Development of a process modelling methodology and condition monitoring platform for air-cooled condensers

Prepared by:

Rashid Ahmed Haffejee

HFFRAS002

Department of Mechanical Engineering

University of Cape Town

Supervisor:

Dr Ryno Laubscher

December 2020

Submitted to the Department of Mechanical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Master's of Science degree in Mechanical Engineering

Key Words: Air-cooled condenser; Dry-cooling; 1-D thermofluid network modelling; Two-phase flow; Machine learning; Data-driven surrogate modelling; Neural networks;

Abstract

Air-cooled condensers (ACCs) are a type of dry-cooling technology that has seen an increase in implementation globally, particularly in the power generation industry, due to its low water consumption. Unfortunately, ACC performance is susceptible to changing ambient conditions, such as dry bulb temperatures, wind direction, and wind speeds. This can result in performance reduction under adverse ambient conditions, which leads to increased turbine back pressures and in turn, a decrease in generated electricity. Therefore, this creates a demand to monitor and predict ACC performance under changing ambient conditions.

This study focuses on modelling a utility-scale ACC system at steady-state conditions applying a 1-D network modelling approach and using a component-level discretization approach. This approach allowed for each cell to be modelled individually, accounting for steam duct supply behaviour, and for off-design conditions to be investigated. The developed methodology was based on existing empirical correlations for condenser cells and adapted to model double-row dephlegmators. A utility-scale 64-cell ACC system based in South Africa was selected for this study. The thermofluid network model was validated using site data with agreement in results within 1%; however, due to a lack of site data, the model was not validated for off-design conditions. The thermofluid network model was also compared to the existing lumped approach and differences were observed due to the steam ducting distribution.

The effect of increasing ambient air temperature from $25°C - 35°C$ was investigated, with a heat rejection rate decrease of $10.9\,MW$ and a backpressure increase of $7.79\,kPa$ across the temperature range. Condensers' heat rejection rate decreased with higher air temperatures, while dephlegmators' heat rejection rate increased due to the increased outlet vapour pressure and flow rates from condensers. Off-design conditions were simulated, including hot air recirculation and wind effects. For wind effects, the developed model predicted a decrease in heat rejection rate of $1.7\,MW$ for higher wind speeds, while the lumped approach predicted an increase of $4.9\,MW$.

For practicality, a data-driven surrogate model was developed through machine learning techniques using data generated by the thermofluid network model. The surrogate model predicted system-level ACC performance indicators such as turbine backpressure and total heat rejection rate. Multi-layer perceptron neural networks were developed in the form of a regression network and binary classifier network. For the test sets, the regression network had an average relative error of 0.3%, while the binary classifier had a 99.85% classification accuracy. The surrogate model was validated to site data over a 3 week operating period, with 93.5% of backpressure predictions within 6% of site data backpressures. The surrogate model was deployed through a web-application prototype which included a forecasting tool to predict ACC performance based on a weather forecast.

Declaration

I, Rashid Ahmed Haffejee, hereby declare the work contained in this dissertation to be my own. All information which has been gained from various journal articles, textbooks or other sources has been referenced accordingly. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor

Signed by candidate

2020-09-18

R.A Haffejee

Date

Acknowledgements

In the name of Allah SWT, the Most Gracious, the Most Merciful.

I would like to thank my supervisor, Dr Ryno Laubscher, for his continuous support, mentorship, patience, and feedback over the last two years, as well as for the various opportunities to publish my research. I would also like to thank you for guiding me to follow a path in research.

I would like to thank the Eskom Power Plant Engineering Institute (EPPEI) specialisation centre for Energy Efficiency, the Applied Thermofluid Process Modelling (ATProM) research group, the National Research Foundation (NRF) [Grant Number 122957], and the Reino Stegen Scholarship donors for funding this project.

Additionally, I would like to thank all my friends and colleagues at the ATProM office and UCT for their support during the project. Special thanks to Priyesh Gosai for his insight and support throughout the project and beyond. I would also like to thank the Department of Mechanical Engineering for their assistance throughout the degree.

Importantly, I would like to thank my parents and siblings, for their unwavering and invaluable support and their continuous encouragement to go the step further, which I will always cherish.

Table of Contents

List of Figures

List of Tables

List of Nomenclature

Acronyms and Abbreviations

ACC	Air-cooled condenser
ANN	Artificial neural network
API	Application programming interface
AWS	Amazon web services
BC	Binary classifier
C	Condenser
CE	Cross-entropy
CFD	Computational fluid dynamics
CSP	Concentrated solar power
DALR	Dry adiabatic lapse rate
DDS	Data-driven surrogate
DNN	Deep neural network
DOE	Design of experiments
EES	Engineering equation solver
GUI	Graphical user interface
LHS	Latin hypercube sampling
M	Mixed Cell
ML	Machine learning
MLP	Multi-layer perceptron
MSE	Mean squared error
NTU	Number of transfer units
R	Regression
ReLU	Rectified linear unit
SDD	Steam distribution duct
SGD	Stochastic gradient descent
SVM	Support vector machine

List of Nomenclature for Thermofluid Network Modelling

General symbols

A_c	Fan casing area	$[m^2]$
A_e	Fan inlet area	$[m^2]$
A_{fr}	Frontal area of one tube bundle	$[m^2]$
D	Tube diameter	$[m]$
$DALR$	Dry adiabatic lapse rate	$[K/m]$
H	Enthalpy, Height	$[J/kg]$, $[m]$
H_0	Specific stagnation enthalpy	$[J/kg]$
H_{HEmid}	Elevation at heat exchanger tube midpoint	$[m]$
H_{top}	Elevation at the top of the A-frame	$[m]$
K_c	Inlet header contraction loss coefficient	
K_{do}	Flow downstream obstacle loss coefficient	
K_{Fs}	Fan static pressure rise coefficient	
K_{in}	Inlet header total loss coefficient	
K_{rec}	Pressure recovery coefficient	
K_{ts}	Heat exchanger inlet support loss coefficient	
K_{up}	Flow upstream obstacle loss coefficient	
$K_{\theta t}$	Total loss coefficient for non-isothermal flow	
L	Tube length	$[m]$
Ny	Characteristic heat transfer parameter	$[m^{-1}]$
Pr	Prandtl number	
Q	Volume flow rate	$[m^3/s]$
$\dot{Q}$	Heat transfer rate to fluid	$[W]$
Re_{Vin} , Re_{Vout}	Vapour Reynolds number at tube inlet/outlet	
Re_{vn_i}	Reynolds number for vapour flow towards condensing surface	
Ry	Air-side characteristic flow parameter	$[m^{-1}]$
T_{ai}	Inlet air temperature to tube	$[m]$
T_{vm}	Mean vapour temperature	$[K]$

UA_{C_i}	Approximate overall heat transfer coefficient based on condensing surface area	$[W/K]$
V	Velocity	$[m/s]$
~~V~~	Volume	$[m^3]$
W_t	Tube width	$[m]$
a_1 , a_2	Frictional coefficients	
b_1 , b_2	Heat transfer parameter coefficients	
c_{pa}	Specific heat of air	$[J/kg \cdot K]$
d_e	Tube hydraulic diameter	$[m]$
h	Heat transfer coefficient	$[W/m^2 \cdot K]$
i_{fg}	Latent heat of vaporization	$[J/kg]$
m_{af}	Air mass flow rate on one side of a finned tube	$[kg/s]$
n_{tb_1} , n_{tb_2}	Number of tubes per bundle in row 1 and row 2	
p_0	Stagnation pressure	$[Pa]$
$p_{a_{BC}}$	Air-side pressure boundary condition at the top of the A-frame	$[Pa]$
Δp_{fric}	Frictional pressure drop through tube	$[Pa]$
$\overline{x}$	Quality	

Greek symbols

α	Crank-Nicholson time-step weighting factor	
α_H	Average homogenous volume fraction	
σ	Inlet header contraction loss factor	
σ_{34}	Exit header loss factor	
θ	Half the A-frame apex angle	$[°]$
ζ	Thermal conductivity, Specific heat (constant pressure)	$[W/m \cdot K]$, $[J/kg \cdot K]$

Superscripts

C	Condensate
H	Homogenous mixture
V	Vapour

a	Air-side properties
i	Heat exchanger tube row index for row 1 or 2
t	Tube row properties
a_{ground} , $a_{FanInlet}$, a_{HEout}	Air-side properties at ground level, fan inlet, and outlet from heat exchangers

Subscripts

o	Property at previous time-step

List of Nomenclature for Data-driven Surrogate Modelling

General symbols

C	Cost function
C_{CE}	Cross-entropy cost function
C_{MSE}	MSE cost function
L	Total number of layers in MLP network
X	Inputs from target dataset
Y	Outputs from target dataset
a_j^l	Output of j-th neuron on the l-th layer
b_j^l	Bias term for the j-th neuron in layer l
d_l	Number of neurons in layer l
k	Number of input parameters
l	Current layer in MLP network
$\overline{m}$, $\overline{v}$	First and second moment vector for Adam algorithm
n	Number of data samples
w_{kj}^l	kj-th weight parameter linking the k-th neuron in layer $l-1$ with the j-th neuron in layer l
$\hat{y}$	Predictions vector from MLP network
z_j^l	Linear component from j-th neuron in layer l. Equivalent to output from j-th neuron before being passed through the activation function

Greek symbols

β_1 , β_2	First and second decay hyperparameter for Adam algorithm
η	Learning rate hyperparameter
δ_j^l	Working gradient of cost function with respect to linear component for j-th neuron in layer l
μ	Mean
σ	Standard deviation
$\sigma_l()$	Activation function for layer l
$\sigma_{Linear}()$	Linear activation function
$\sigma_{ReLU}()$	ReLU activation function

$\sigma_{Sigmoid}()$	Sigmoid activation function
$\hat{\theta}$	Trained parameters including weights and biases

Subscripts

t	Current iteration used in Adam algorithm

Publications from current work

1. R. A. Haffejee and R. Laubscher, “Development of a thermofluid network modeling methodology for double-row air-cooled condensers”, *Therm. Sci. Eng. Prog.*, vol. 19, Oct. 2020.

2. R. A. Haffejee and R. Laubscher, “Application of machine learning to develop a real-time air-cooled condenser monitoring platform using thermofluid simulation data”, *Energy and AI.* (submitted, under revision)

1. Introduction

1.1 Background and Motivation

Cooling technologies have become paramount to how the world functions today. From the generation of electricity to cooling offices and homes, cooling technologies are one of the most significant technological developments of the past century. Cooling is also the crux of many industrial systems, such as in power plants as well as data centres, making it a key design component.

With societal calls to be more environmentally conscious arising globally, new regulations have been adopted, such as the Paris Agreement battling climate change, the Kigali Amendment, and Kigali Cooling Efficiency Program (K-CEP). As a result, cooling technologies are required to be more efficient as well as use fewer resources to provide cleaner energy [1], [2]. This is further compounded by the increase in global population sizes along with rising energy demands, motivating the need to research and develop sustainable cooling technologies, particularly in an energy generation context.

Dry cooling is of particular importance, especially in the context of power generation systems. Dry cooling involves cooling the power cycle working fluid using ambient air and indirect contact heat exchangers. Cooling of the working fluid in a power cycle is necessary to satisfy the Kelvin-Planck statement [3]. Therefore, the heat rejection process in a power cycle is facilitated by the cooling system. For example, in Brayton cycles, the working fluid is cooled before it enters the compressors. In Rankine cycles, the saturated water vapour leaving the turbines are condensed before it is fed to the plant pumps using the dry cooling system. The primary benefit of dry cooling technologies is the minimal amount of water consumed. The adoption of direct dry cooling technology in Rankine power cycles, known as an Air-Cooled Condenser (ACC), has exponentially increased from 1992-2007, shown in Figure 1 [4].

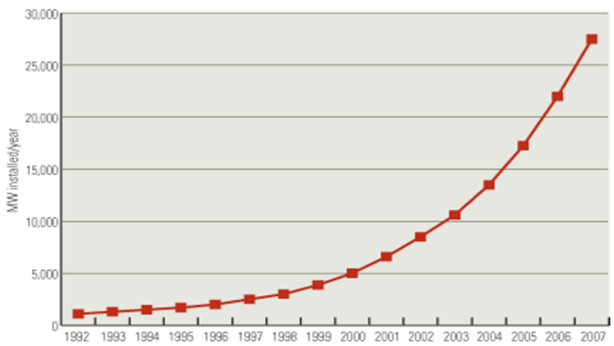


Figure 1: Dry Cooling Installed Global Capacity (MW) from 1992-2007 [4]

Dry cooling's counterpart, wet cooling, generally uses cooling towers, which combine heat and mass transfer to condense the working fluid. A separate cooling circuit loop is used to condense the process fluid, steam, while air is used to cool the cooling circuit loop. Around one to three per cent of circulating cooling water is lost to evaporation [5]. In a typical fossil-fuelled power plant, approximately 1.0 – 2.0 litres of cooling water are required per *kWh(e)* of generation. In context, total make-up water requirements for a 600 *MW(e)* coal-fired plant with a 70% capacity factor can exceed $11 \times 10^6\ m^3$ per year. For comparison to a dry-cooled system, water usage is reduced by around 90%, consuming around 0.1 litres of water per *kWh(e)* in the same plant [6]. The reduction in water usage is a significant benefit, especially considering environmental regulations, with the World Economic Forum listing water scarcity as the 4th most considerable global risk in terms of impact [7]. Moreover, the plant efficiency against water withdrawal amounts can be seen in Figure 2, with dry air cooling using the least amount of water [8].

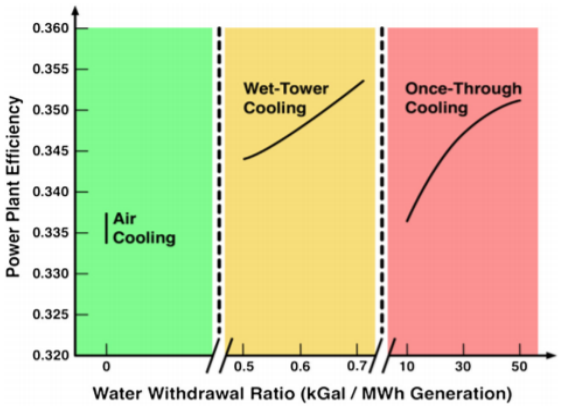


Figure 2: Predicted Efficiency of a 500 MW power plant with dry and wet cooling systems at varying withdrawal rates [8]

It is, however, essential to note the drop in overall generation efficiency using a dry-cooled system compared to a wet-cooled one, also shown in Figure 2. This efficiency difference is primarily due to water being a better cooling fluid than air, as water has a higher thermal conductivity and specific heat capacity. The increased efficiency in water cooling reduces the condenser pressure and therefore results in a greater increase in the total enthalpy change through the turbines compared to air cooling. Furthermore, dry cooling systems are more susceptible to varying ambient conditions than wet cooling alternatives, particularly in direct dry-cooled systems, where air properties directly influence cooling capacity. Dry-cooling systems also require more initial capital, due to substantial manufacturing and assembly costs, as well as a significant number of unit installations to meet the drop in efficiency [9]. An existing ACC installation is seen in Figure 3 [10] for scale.

Figure 3: ACC Installation at a power plant in South Africa [10]

On the other hand, where wet cooling generally requires a large body of water located close to the system, dry cooling is not limited in this regard. Dry cooling allows for greater flexibility and deployment in arid regions. This elevates dry-cooling systems as a potential choice for cooling systems used in concentrated solar power (CSP) plants located in regions with higher solar irradiance. With the incorporation of more renewable energy into the grid, CSP plants are aided by a waterless cooling solution. Combined-cycle gas turbines can also benefit from dry cooling, as direct dry cooling can be used to condense the steam in the heat recovery steam generator (HRSG) subsystem without requiring large amounts of water, lifting restrictions on plant location.

In a South African context, where energy security is becoming increasingly important, the need to efficiently design and optimise power plants is crucial. With the national electricity provider, Eskom, using dry-cooling with ACCs in four power plants, there is a need to monitor and predict ACC performance under varying ambient conditions, seeing as it has a direct impact on the power cycle thermal efficiency.

The increasing relevance of dry cooling and its implementation has been shown. The need for research and development of dry cooling is growing, driven mainly by limitations in system efficiency, and the need to optimise and improve cooling performance to match that of wet cooling while still retaining the environmental benefits.

1.2 Problem Statement and Research Objectives

This study focuses primarily on ACCs used in Rankine power cycles. ACC monitoring is quite limited, with the turbine backpressure used as a critical performance metric. Unfortunately, ACC performance deteriorates under adverse ambient conditions, such as crosswinds and high ambient

dry-bulb temperatures, on top of the already reduced efficiency compared to wet cooling. The reduced efficiency results in more considerable capital costs, as the system must be oversized to compensate for reduced performance during adverse ambient conditions. These factors act as barriers to ACC adoption. In the present work, a detailed 1-D steady-state systems-level thermofluid simulation model of a utility-scale ACC system is developed and used to generate data for a machine learning based monitoring platform. The presented modelling methodology is applied to an actual case study of a utility-scale power generation cycle cooling system.

Due to the massive variation in ACC performance with ambient conditions, it is vital to be able to monitor these systems. As the ambient temperature changes throughout the day, the ACC heat rejection rate and consequently backpressure also varies. These changes become even more complicated when considering crosswinds that develop, affecting the performance significantly. Therefore, this creates a demand to monitor and predict ACC performance under varying ambient conditions accurately. Monitoring would allow plant operators to gain insight into ACC performance based on forecasted weather conditions and accurately estimate expected plant energy output.

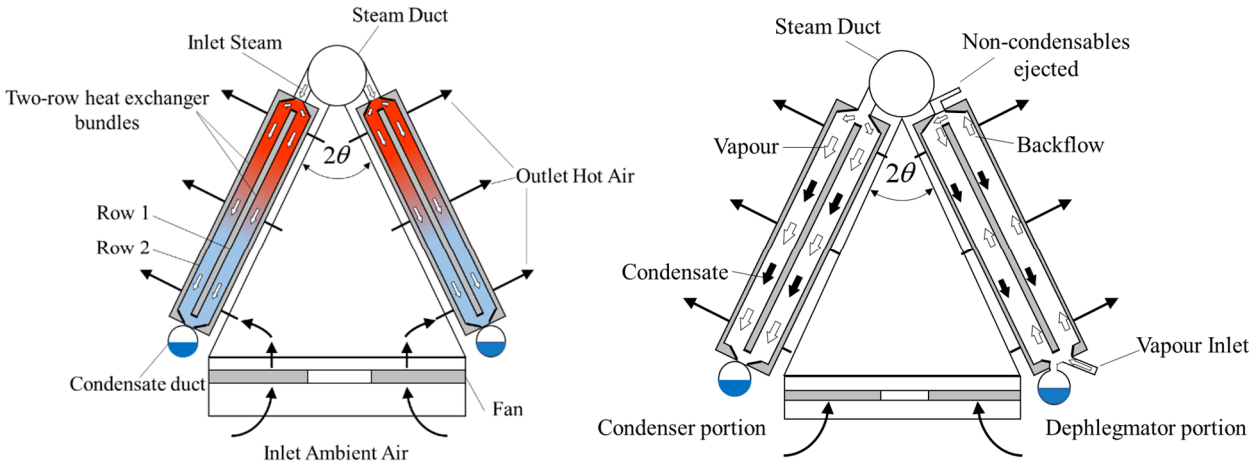


Figure 4: Overall layout of a two-row A-frame condenser cell (left) and mixed cell (right)

An ACC system typically consists of two different condensing heat exchangers, namely a condenser and a dephlegmator. Condensers make up most of the cooling system and are where primary condensation occurs. Condensers receive inlet steam from the main steam distribution ducts (SDDs) which is fed, in turn, by the turbine exhaust duct. Dephlegmators act as secondary condensers and are used to condense the remaining steam flowing out of the condenser cells and to eject any non-condensable gases (air) trapped in the fluid stream. Typically, in ACC systems, cells are defined as bundles of heat exchangers which are fed by a single axial flow fan. A typical two-row A-frame

condenser cell is shown on the left of Figure 4. For a double row heat exchanger, considered in this research, steam splits at the row inlet into two individual heat exchanger rows. An axial fan located at the base of the cell forces air through the heat exchanger rows. Consequently, the second row receives hotter inlet air which has already passed through the first row.

Dephlegmators can operate in mixed cells, where a specific portion of the cell's heat exchanger bundles acts as a condenser and the remaining portion as a dephlegmator. The mixed cell has a similar layout, however, for the dephlegmator portion, vapour enters the heat exchangers from below and travels up into the heat exchangers tubes. The vapour flow is driven by the decrease in volume of water when condensed into steam, creating a suction effect. Non-condensable gases trapped in the vapour stream also exit through the ejector. The overall layout for a mixed cell is shown on the right of Figure 4. For the dephlegmator portion, vapour from the condenser cells flows into the dephlegmator portion from the bottom of the heat exchanger rows. Vapour flows in the opposite direction to the condensate, as the condensate falls due to gravity. Additionally, backflow could occur between the two rows, where vapour does not fully condense and thus enters the preceding row at the top of the dephlegmator heat exchanger. This phenomenon is not an issue in single row ACCs, as steam would not enter a second row. Non-condensable gases trapped in the vapour stream are ejected at the top of the dephlegmator half.

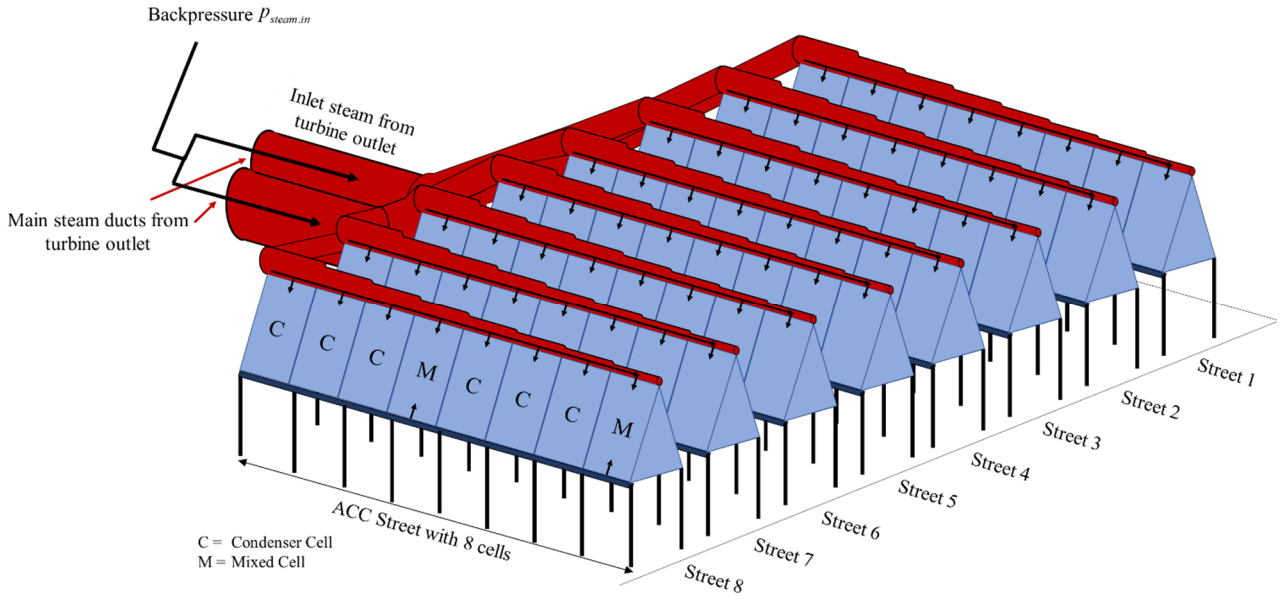


Figure 5: Typical ACC system receiving inlet steam from main steam ducts

Figure 5 shows the typical layout of an ACC system. Steam from the turbine outlet enters the ACC system through the main steam distribution ducts. The steam ducts then split into individual

'streets' of ACC cells. For the case study ACC system in this research, a single street consists of a total of eight cells; six condenser cells, and two mixed cells, as shown in their respective order in Figure 5. The case study system has a total of eight streets. Each condenser cell and mixed cell in the street are as shown previously in Figure 4.

Network modelling or process modelling is an important tool that helps model complex systems such as utility-scale ACCs found in power generation. 1-D network modelling allows for different properties of the system to be analysed at discretised points in the flow stream and helps to characterise flow development to simulate the relevant pressure drops and flow paths through the system over a range of operating conditions. A cell-by-cell discretisation for a utility-scale ACC system allows for each cell to be modelled. A case study of an existing 64-cell ACC system in South Africa was modelled in the present work. In comparison to 3-D CFD typically used for modelling ACC related phenomena, 1-D process modelling is less computationally expensive, which allows for an entire utility-scale ACC system to be simulated. On the other hand, 1-D network modelling does not capture the complex air-side interactions that occur in ACCs and would need to be coupled to CFD in future work when considering detailed wind effects. Flownex SE® 2019 was used for network modelling.

This research builds on previous thermo-hydraulic ACC modelling methodologies developed by Kröger [5]. Kröger developed a comprehensive methodology for a condenser cell; however, did not consider a dephlegmator or mixed cell. The project builds on this methodology and develops it further considering a mixed cell and dephlegmator heat exchangers. It is important to note that only two-row ACCs were considered. The network model captures the overall ACC thermal-hydraulic performance by solving the 1-D forms of the mass, momentum and energy balance equations along with various component characteristic equations such as pressure drop and convective heat transfer correlations.

Machine learning (ML) is another powerful tool that can be utilised in a condition monitoring context. ML consists of data analysis techniques that identify statistical trends using artificial intelligence and pattern recognition. The usability of an ML model greatly depends on the data that the model is developed upon. Artificial neural networks (ANNs) are a type of ML model that can be used to reduce the computational expense of physics-based simulation models significantly. Typically these ML models are developed using empirical data, but more often than not, in the simulation and analysis of complex engineering systems, the cost of data acquisition is prohibitive [11]. One alternative is to use simulation data to develop these predictive ANN models. ANNs can be trained on data generated by complex physics-based models (the 1-D network model of the ACC) using traditional numerical techniques. The ANNs can thereafter ideally produce similar results as the numerical network model but in a more computationally efficient manner. Deep learning can

also be used with ANNs, also called deep neural networks (DNNs), allowing for more complicated phenomena and non-linearities to be learnt by the neural network. The developed ML models based on the network model can, therefore, be referred to as a data-driven surrogate (DDS) model.

The developed model must ideally be useable in conjunction with the actual ACC system, and with varying ambient conditions, be able to produce solutions rapidly. The use of a DDS model is well-suited to address computational limitations. It is, therefore, key to the development of a condition monitoring platform, demonstrating the need for ML in this research. As the DDS model is dependent on the 1-D network model, it is also crucial to ensure the validity of the 1-D network model. The DDS model can also be coupled to a weather server that forecasts ambient temperature and wind conditions. This forecast will, in turn, allow for prediction of future ACC performance based on upcoming ambient conditions, giving plant operators more insight into future performance.

The main objectives of the research are shown in Figure 6, with the thermofluid network modelling and data-driven surrogate modelling making up most of the project. The web-app prototype was developed as a way of demonstrating the practicality of the data-driven surrogate model, rather than a fully-fledged implementation.

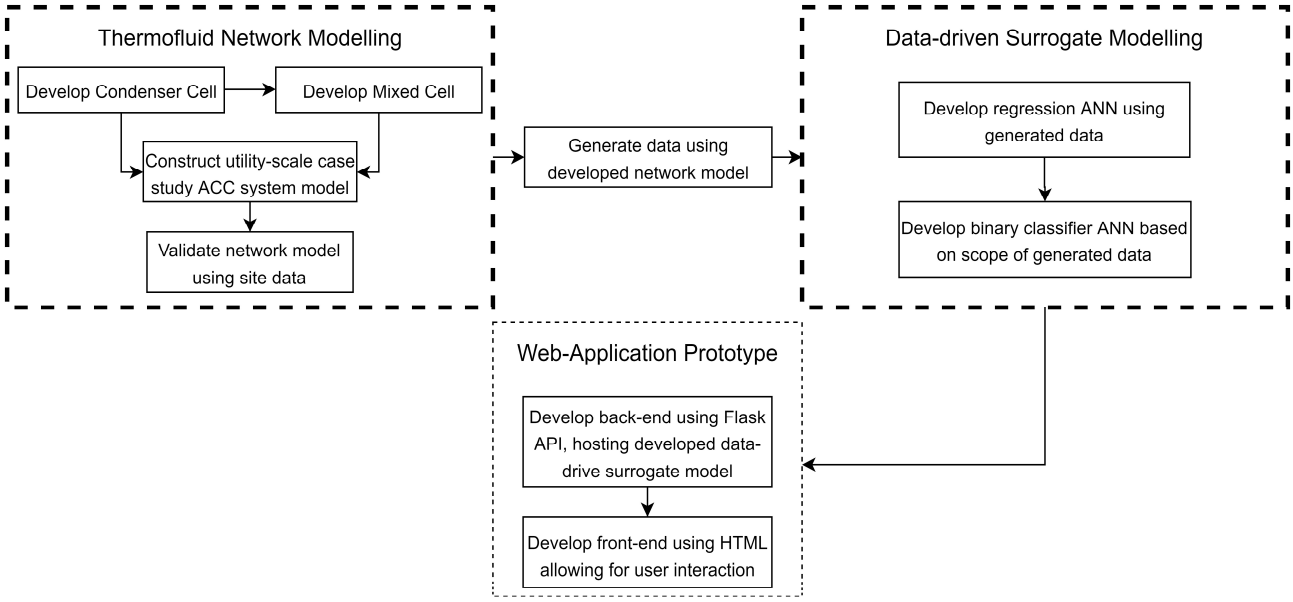


Figure 6: Project Objectives and Breakdown

2. Literature Review

A literature review was conducted in two main sections; ACC modelling, and machine learning modelling.

2.1 Thermofluid Modelling of ACC Systems

ACC modelling is further broken down into existing ACC empirical correlations, computational fluid dynamics (CFD) models, and lastly steam-side and network models.

Kröger [5] developed a thermal-hydraulic modelling methodology for a two-row A-frame condenser cell with elliptical finned tube heat exchangers. Steam-side pressure drops modelled included inlet losses from tube headers to the tube bundles, as well as frictional losses and momentum recovery in the tube bundles. The associated control volume was a single tube row heat exchanger. Heat transfer rates were determined using the effectiveness-NTU method, with heat transfer coefficients for both air-side and tube-side determined empirically. Air-side modelling was solved using a draft equation which balanced pressure losses through the fan assembly and heat exchangers with the pressure rise from the axial fan, solving for the required air mass flow using an iterative solving process. The airflow path was assumed to be uniform through the heat exchangers. This simplification would therefore not account for changes in the airflow distribution across the heat exchangers. The approach presented by Kröger [5] models the entire ACC system as a single condenser and dephlegmator control volume, which solves the mass and energy balance equations. The detailed breakdown of empirical correlations from Kröger used in this research is presented in Chapter 3.

O' Donovan et al. [12] explored the pressure drop due to steam condensation in ACC tube bundles. With the comparison of experimental pressure drops, good agreement was found with two-phase frictional pressure drop correlations, including the Lockhart & Martinelli correlation. Yang et al. [13] investigated the thermal flow characteristics of wave-finned flat-tube bundles and developed correlations for friction factors as well as heat transfer. Additionally, Davies et al. [14] presented a model incorporating inclined stratified-flow condensation on the steam-side with detailed modelling of film condensation as well as condensation pooling using existing empirical correlations.

From the few empirical correlations discussed above, Kröger's [5] was the best suited for a network modelling approach for elliptical finned tube bundles, with both steam-side and air-side correlations developed. Consequently, this could be implemented on a cell-by-cell basis. The other relevant correlations used by Davies et al. [14], was less suited to a network modelling approach, mainly because of associated complexity with the condensation river and pooling in the tube bundles.

As ACCs specifically are affected by crosswinds and ambient temperature due to the higher air-side thermal resistance, the majority of ACC research has revolved around the use of CFD models with less focus on steam-side performance and 1-D network models. In particular, numerical and experimental investigations on fan flow characteristics in ACC systems have been conducted [15]–[18]. Several numerical models for fan performance have been implemented, such as the pressure jump fan model used by Owen [19], the actuator disk model used by Engelbrecht [20], as well as the extended actuator disk model and the reverse-engineered empirical actuator disk model used by Wilkinson et al. [21].

Owen [19] developed an efficient method to evaluate the performance of ACCs under windy conditions using CFD and a pressure jump fan model, attempting to reduce effects of crosswinds through winds screens, walkways, and adjusting fan speeds. A case study power plant at Nevada, USA, was used for the investigation, and it was found that installation of an appropriate windscreen helped reduce the adverse effects of wind on performance.

Engelbrecht [20] developed a CFD model of ACCs using the actuator disk fan model, and determined effects of crosswinds on ACC performance. It was found that ACC performance decreased with increasing wind speeds. Additionally, the effect of wind speed on the air volume flow rate for affected cells was investigated, with a decrease in air volume flow rate observed with increasing wind speeds. Moreover, Meyer et al. [22] noted the presence of kinetic energy recovery on the air-side flow through the plenum. The authors recommended a recovery factor (K_{rec}) of 0.3 for heat exchanger flow loss coefficients (K_{he}) between 15 and 25. Engelbrecht et al. [23] incorporated a K_{rec} of 0.527 compared to that observed by Meyer et al. [22] of 0.553 for the experimentally tested B2-fan and A-frame heat exchanger. Engelbrecht et al. [23] found good agreement between numerical and experimental K_{rec} values after iteratively tuning K_{rec} to match numerical results and the analytical model by Kröger [5]. An over-prediction of numerical results compared to the analytical solution (without recovery) was noted, due to the increased air volume flow rate with the additional kinetic energy recovery. These results can be seen from Figure 7, for a given pressure, the corresponding air volume flow rate with recovery would be higher than without recovery.

Yang et al. [24] investigated the effects of ambient winds and spacing of ACCs, as well as Chen et al. [25], who explored a novel layout of ACCs to improve flow performance under adverse wind conditions. Findings included a difference in performance for cells downstream due to increased ambient air temperatures. A vertical arrangement of cells was found to minimise this effect and consequently resulted in increased ACC performance, and in turn, a lower backpressure.

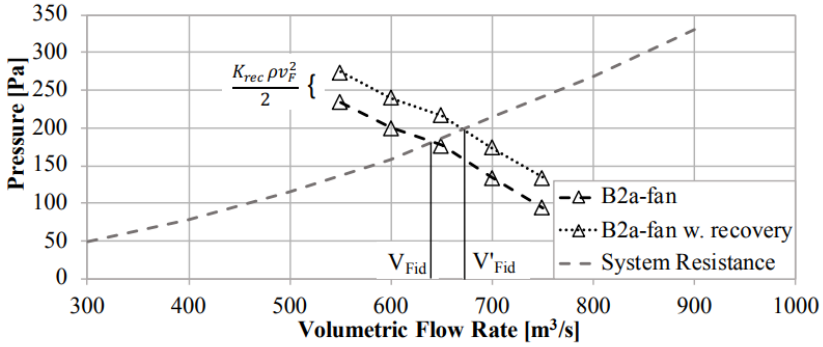


Figure 7: Effect of kinetic energy recovery on pressure rise and air volume flow rate from Engelbrecht et al. [23]

In addition to the CFD models above, Hooman et al. [26] explored porous medium modelling for ACCs. ASPEN B-JAC, a commercial heat exchanger design software, was used to calculate the pressure drop for the tube bundle properties. It was found that empirical and analytical correlations could be used to predict the thermohydraulic performance of the heat exchangers without the need for time-consuming and computationally expensive CFD simulations. Deng et al. [27] conducted a numerical study using CFD on flow characteristics of reflux condensation in ACCs. The authors reiterated the need for operations on both the steam and air sides to be considered to improve ACC performance, along with several other authors, including [12], [14], [28]–[31]. A sensitivity analysis was performed on the complete condensation length and condensate thickness layers during vapour condensation in an up-flow flat tube. A strong correlation was found between the complete condensation length and vapour mass flow rate as well as the inlet vapour Reynolds number.

Considering steam-side modelling, Mahvi et al. [29] explored challenges in predicting the steam-side pressure drops and heat transfer in ACCs. They noted the lack of ACC adoption, particularly in the USA, with ACC adoption at 1% of power plants, mainly due to capital costs and reduced efficiencies. These barriers highlighted the need to develop models to understand ACC performance parameters better, as well as potentially highlight critical areas for design improvements. A model was developed using the Engineering Equation Solver (EES), using the Kröger [5] approach for losses in the A-frame condenser, and the effectiveness and NTU method for the heat transfer rates. The focus was on steam pressure drops along the length of the heat-exchanger tube bundles, assuming a uniform steam-side distribution across the ACC system with a focus on a single cell. Using a range of analytical and empirical formulations, the condensate heat transfer coefficients varied from 1660-24000 $W/(m^2K)$, depending on the correlation used. Additionally, it was found that predicted plant efficiency varied by 1.7% between the different pressure drop correlation models, indicating that both steam-side and air-side effects must be considered to improve ACC performance.

Lin et al. [32] explored improving the air-side heat transfer performance in an identical setup as Mahvi et al. [29]. Performance was quantified under a constant ambient air temperature with different fin configurations. The model was also integrated into a Rankine and combined cycle power plant model to determine the impact of fin designs on overall cycle efficiency.

A similar study was conducted by Bustamante et al. [8], investigating the possibility of achieving water-cooled power plant performance using air-cooled condensers. EES software was also used for the modelling of a $500\,MW$ plant. Several pressure-drop correlations were employed, and the findings also echoed those by Mahvi et al. [29], where the relevant correlation used affected plant efficiency. They concluded that to aid heat transfer, the heat transfer coefficient could be increased through several methods, such as mechanical mixing and jet impingement, used in hybrid systems.

Heyns [33] also explored the performance characteristics of a hybrid dry/wet dephlegmator, applied using Kroger's [5] 1-D thermohydraulic modelling approach. The authors found that at high ambient temperatures, the hybrid dephlegmator achieved the same turbine performance as an oversized ACC system or an ACC system with spray cooling, at considerably lower costs.

Gadhamshetty et al. [34] developed a process model to improve ACC performance at combined cycle power plants. The ACC component used a high-level approach connected to other components in the combined cycle plant. A chilled water thermal energy storage system (TES) and absorption refrigeration system (ARS), powered by waste heat, precooled inflow air to the ACC when ambient temperatures were high. The EES software was used to model the TES. A limited high-level ACC model was used in the system model of the closed cycle power plant, observing the backpressure and heat rejection rate as the key parameters, with the main focus on the TES and ARS systems. With a tank volume of 4500 m^3, the inlet air could be kept at $20°C$ throughout the year. Consequently, this allowed for the backpressure to be relatively controlled with varying ambient air temperatures; however, without considering wind effects.

Owen et al. [31] conducted a numerical investigation of vapour flow in large ACCs. The authors noted the importance of numerical models to analyse the nature of performance losses, resulting in adequate condenser cell and dephlegmator cell sizing. Notably, the authors mentioned that the use of CFD to model these large systems was too computationally expensive and time-consuming. Through the combination of CFD data for tube inlet loss coefficients through steam ducting, and a numerical flow distribution code in an iterative solver, a numerical simulation method for vapour flow distribution in ACC tube bundles was developed. This simulation method allowed for the effects of steam maldistribution in cells along an ACC street to be modelled, with varying inlet header loss coefficients. Due to the maldistribution, an additional demand was placed on the dephlegmator, with an additional 6% vapour flow through condensers required to prevent back-flow.

Davies III et al. [14] also developed a thermal-hydraulic model based on a $10.7\,m$ tube bundle and validated with experimental procedures on a $5.7\,m$ tube. On the steam side, the model separated flow into two sections with steam flowing at the top of the tube and a condensate river along the bottom. On the air-side, the heat transfer coefficient was predicted using a combination of empirical models and CFD. Kröger's [5] correlations were also tested with the experimental procedure, and overpredicted capacity by 3% on average, with an experimental uncertainty of 3%. The main aim of the thermal-hydraulic model was to investigate complex void fractions, capacity, and overall heat transfer coefficient in a single flattened-tube.

Klimes et al. [35] designed a semi-empirical model of A-frame ACCs as an alternative tool to CFD based models. The model was discretised using 2-D control volumes for the tube bundles, which incorporated existing empirical correlations from literature and consisted of three sub-models, which were the steam-side, air-side, and fan. The model had four tube rows; a single dephlegmator row while the rest were condenser rows. A single set of tube bundles were modelled and scaled to represent a single cell or module. Consequently, off-design conditions such as hot air recirculation or wind effects were not modelled. The typical simulation time was between 15-30 minutes, which was relatively low compared to more expensive CFD simulations. Validation was completed using manufacturer datasheets as well as experimental data at a waste incineration plant.

The existing steam-side ACC models in literature mainly focus on steam-side pressure drop correlations in the tube bundles or uniform high-level models in conjunction with other power plant subsystems. Developing a sufficiently detailed ACC model that models steam-side properties on a cell-by-cell basis and in turn row-by-row for an entire ACC system can help determine the effect of various boundary conditions on system performance. These include ambient temperatures, hot air recirculation effects, as well as wind effects. Moreover, vapour distribution into cells along a street (axially), has shown to affect ACC performance as shown by [31]. A cell-by-cell discretisation approach allows for the vapour duct distribution effects to be considered.

The two types of condensing heat exchangers (condensers and dephlegmators) both have different performances and functions, and it is vital to simulate both for a representative model. A steam-side dephlegmator model in literature has only been observed in [36], which modelled only a single row of dephlegmator tube bundles. Additional studies related to dephlegmators included an experimental setup for a hybrid dry/wet dephlegmator [33] and the effect of vapour flow distribution in condensers on dephlegmators [31].

2.2 Machine Learning Modelling

The current literature study on machine learning modelling is split into two sections, namely; data-driven surrogate models, and data-driven models in the energy generation context. Data-driven surrogate modelling through machine learning is a relatively new and upcoming research field. The field requires in-depth knowledge of both the physical system being modelled, as well as knowledge of machine learning theory to develop the surrogate model, making the research topic relatively specialised. Consequently, literature surrounding the topic was quite limited, especially in the energy generation context. Nonetheless, a brief overview of the literature in data-driven surrogate modelling is presented, followed by a broader overview of data-driven models (typically using experimental data) in the context of power generation.

Liang et al. [37] investigated the use of deep neural networks (DNNs) to predict the hemodynamics of human thoracic aorta. The research was driven by the need for quick computational times and more straightforward procedures for patient-specific analysis. Data from CFD simulations were used to train the networks. The DNNs take the shape of the aorta as input and provide hemodynamic distributions within one second of computation time. The model had an average error in velocity magnitude of 1.96%, displaying similar results with the CFD simulations but at a fraction of the computing expense.

Data-driven surrogate modelling has also been used in aerodynamic analysis on aerofoils, such as by Wu et al. [38]. A deep learning approach using generative adversarial networks (GANs) and convolutional neural networks (CNNs) was developed to predict 2D flow characteristics around an aerofoil. A training set of 500 aerofoils was used to train the model. The model used a one-to-one mapping of aerofoil geometry characteristics to define the 2D pressure field associated with the respective aerofoil. The model predicted the flow field image within a few seconds, as compared to several hours for typical CFD simulations. In turn, aerodynamic characteristics could then be extracted from the flow field pattern to be used in design space exploration.

Warey et al. [39] used machine learning (using an ANN) and CFD simulations to predict vehicle cabin thermal comfort. An existing validated CFD model of a vehicle cabin was used to generate data to train the machine learning model. Hyperparameter optimisation was conducted, including the number of hidden layers, neurons per hidden layer, activation functions, batch sizes, and more. Ten-fold cross-validation was used to select the optimal network. The machine learning model was able to predict the cabin air temperature as well as the equivalent homogenous temperature for each passenger with a <5% test error. The low test error highlighted the potential robustness of data-driven surrogate modelling without having to rely on expensive CFD simulations for each use case.

Shi et al. [40] explored the optimization of combustion in an ultra-supercritical boiler using machine learning techniques. ANNs were used for predicting boiler operating and emission properties. The ANN model received the unit load, coal properties and excess air, and predicted the thermal efficiency and NOx emissions. CFD simulation data, in conjunction with experimental data, formed the dataset used to train the ANN. Additionally, a genetic algorithm was used to optimise for higher thermal efficiency and reduced NOx emissions. The model predictions achieved mean errors of 0.04% for thermal efficiency, indicating the surrogate model predicted results in agreement with the dataset.

In the context of data-driven models based solely on experimental data, Fast et al. [41] investigated the application of artificial neural networks (ANNs) for on-line condition monitoring and diagnosis of a combined heat and power plant. ANNs were constructed for each component in the plant at steady-state operation. Data covering three months of plant operation was used to train the individual ANNs. Average prediction accuracies for the various ANNs were <3%. Moreover, a GUI was developed to host the ANNs on-line in conjunction with the plant monitoring system, demonstrating the practicality of the data-driven models. If model predictions deviated too far from normal operating conditions, the GUI was used to convey a warning to plant operators for fault checking.

Neural networks have also been used for time-series forecasting specific parameters, such as reheater metal temperatures through the use of recurrent neural networks by Laubscher [42]. A 290 *MWe* coal-fired boiler was used as a case study, with operational data used as input sequences. A sequence of 5 minutes in the future was predicted based on the previous 8 minutes provided to the model. The model had a <1% error on the test set. Additionally, Dhanuskodi et al. [43] used ANNs to predict wall temperatures in supercritical boilers. Experimental data was used to train the model, while the test set consisted of data from literature. The model architecture consisted of four input neurons with a single output. Two hidden layers were used, with ten neurons and five neurons, respectively. An accuracy of 81.94% with a deviation of $\pm 7°C$ was achieved on the test set.

In a cooling context, Wang et al. [44] modelled a hybrid ejector air conditioning system using ANNs. Several machine learning models were investigated, including ANNs and support vector machines (SVMs). Component level ANN models were used, with a 10% average relative error. Several input parameters were sent to the ANN, including backpressure, component pressures and temperatures as well as ambient temperatures. The result from the model was the predicted coefficient of performance at steady-state.

In terms of data-driven models for ACCs, Li et al. [45] developed a data-driven model for ACCs using an SVM. The input variables used were the fan speed, ambient air temperature, wind speed, and exhaust steam mass flow rate and enthalpy from the turbine outlet. The output variable was the

backpressure. Notably, the model lacked flexibility for off-design conditions because of a lack of data availability as well as variation in ACC performance over time due to fouling. An average absolute error of 0.680 *kPa* was achieved on the test set, with test set performance observed in Figure 8.

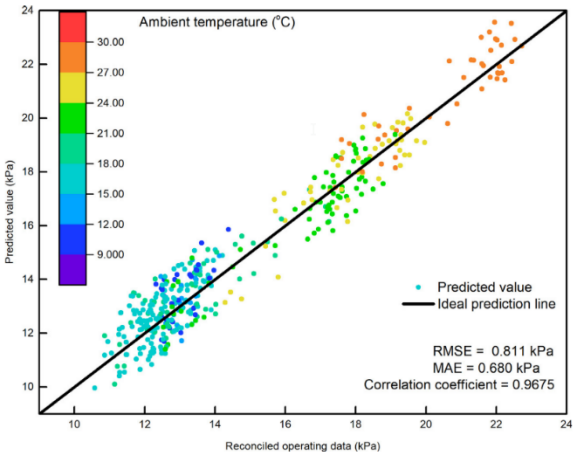


Figure 8: Predicted vs test set backpressures from the developed SVM by Li et al. [45]

Du et al. [46] investigated back pressure predictions using an ANN for ACCs at a 600 *MW* power plant with 56 ACC cells. In contrast to Li et al. [45], the ANN was also trained on data under varying ambient weather conditions, accounting for wind and air humidity. Input parameters, in addition to those used by Li et al. [45], included wind direction relative to the ACC system as well as air humidity and the weather status. The ANN used these input parameters to predict the backpressure only. The dataset included data from three seasons, spring, summer, and autumn, as well as varying unit loads and wind conditions. A three-layer network with 22 hidden nodes was selected as the optimal network architecture, with a mean relative error of 9.3% achieved on the test set. The ranges in the dataset for several measured variables can be seen in Table 1. The developed model would be limited by the availability of data, uncertainties in measurements, and data sparsity, especially as only 249 input-output pairs were used. Extrapolating to data ranges not within the dataset used would result in significant errors in predictions, which is a limitation to data-driven models in general.

Table 1: Experimental data ranges used by [46]

Measured Variable	Variable Range	Units
Unit Load	199-657	*MW*
ACC Wind Direction	5.7-98.3	°
ACC Wind Velocity	1.94-5.19	*m / s*
Ambient Air Temperature	-2.24-43.53	*°C*
Turbine Backpressure	9.11-41.88	*kPa*

Data-driven surrogate models were limited in literature; however, applications were observed in several fields. The use of a surrogate model to generate data was shown to be beneficial to capture performance under a wide range of operating conditions, including where experimental data may not be available. Moreover, the development of surrogate models reduced computational expense significantly as compared to typical CFD modelling, which helped make the developed solutions more practical.

The data-driven ML models using experimental data demonstrate that ML is also a tool that can be leveraged in condition monitoring, which helps to develop predictive models for complex non-linear systems. The use of ML in power generation has been shown to range from boiler performance to individual component performance in a power plant, and to ACCs. In the context of ACCs, data-driven surrogate modelling has not been observed in literature. The use of data-driven surrogate models for ACCs may be beneficial as a physics-based model would be used for data generation, in contrast to obtaining experimental data with a potentially limited scope.

3. ACC Thermofluid Modelling: Materials and Methods

The objective of the thermofluid modelling was to develop and implement 1-D thermofluid network modelling methodologies for double row A-frame condenser and mixed cells. Modelling each individual cell in the ACC system required a 1-D modelling approach as typical 3-D CFD models would be too computationally expensive to model the entire system (air- and steam-side). Using Kröger's [5] empirical correlations and methodology for a condenser cell, and a newly developed methodology for the dephlegmator, a sufficiently accurate modelling methodology was established. Consequently, a model based on the case study ACC was developed.

The developed model was compared to the existing lumped 0-D approach based on Kröger's [5] methodology, which solves for single condenser and mixed cell control volumes and is linearly scaled up to represent the entire ACC system. The linear scaling makes the existing lumped approach unsuitable for modelling off-design conditions, particularly wind effects, as these conditions affect multiple cells non-uniformly in the system. This is also observed with steam-side models in literature which were not suitable for off-design modelling, such as [12], [29], [32], [47]. The use of a 1-D network modelling methodology in this research, where each cell in the ACC system was modelled, allowed for these effects to be investigated. Moreover, this also allowed for the steam supply ducting behaviour to be considered. Consequently, results, including heat rejection rates per row, fan power, and air-side properties, were available on a cell-by-cell basis. Non-condensable gases were not considered in the modelling process; however, its presence would reduce vapour partial pressures, and in turn, lead to a slightly reduced heat rejection rate.

3.1 Case study ACC

An entire ACC system was modelled after an existing ACC system for an $800\,MWe$ supercritical unit at a power plant in South Africa, including 64 ACC cells in total with eight streets, as seen in Figure 5. Inlet steam from the turbine exhaust flowed through two main steam distribution ducts, which then split into four streets each, to a total of eight streets.

For this section, the total amount of inlet steam to the ACC system from the two main steam distribution ducts (SDDs) was fixed at $422\,kg/s$ with an inlet quality of 0.9245, and an adjustable inlet vapour pressure (backpressure) $p_{steam.in}$. It is important to note that the total inlet steam flow rate as well as quality could be varied but was fixed to limit the scope of this section. This

backpressure would change depending on ambient conditions as well as the unit load. At steady-state, this backpressure would ensure all steam entering the ACC system would condense.

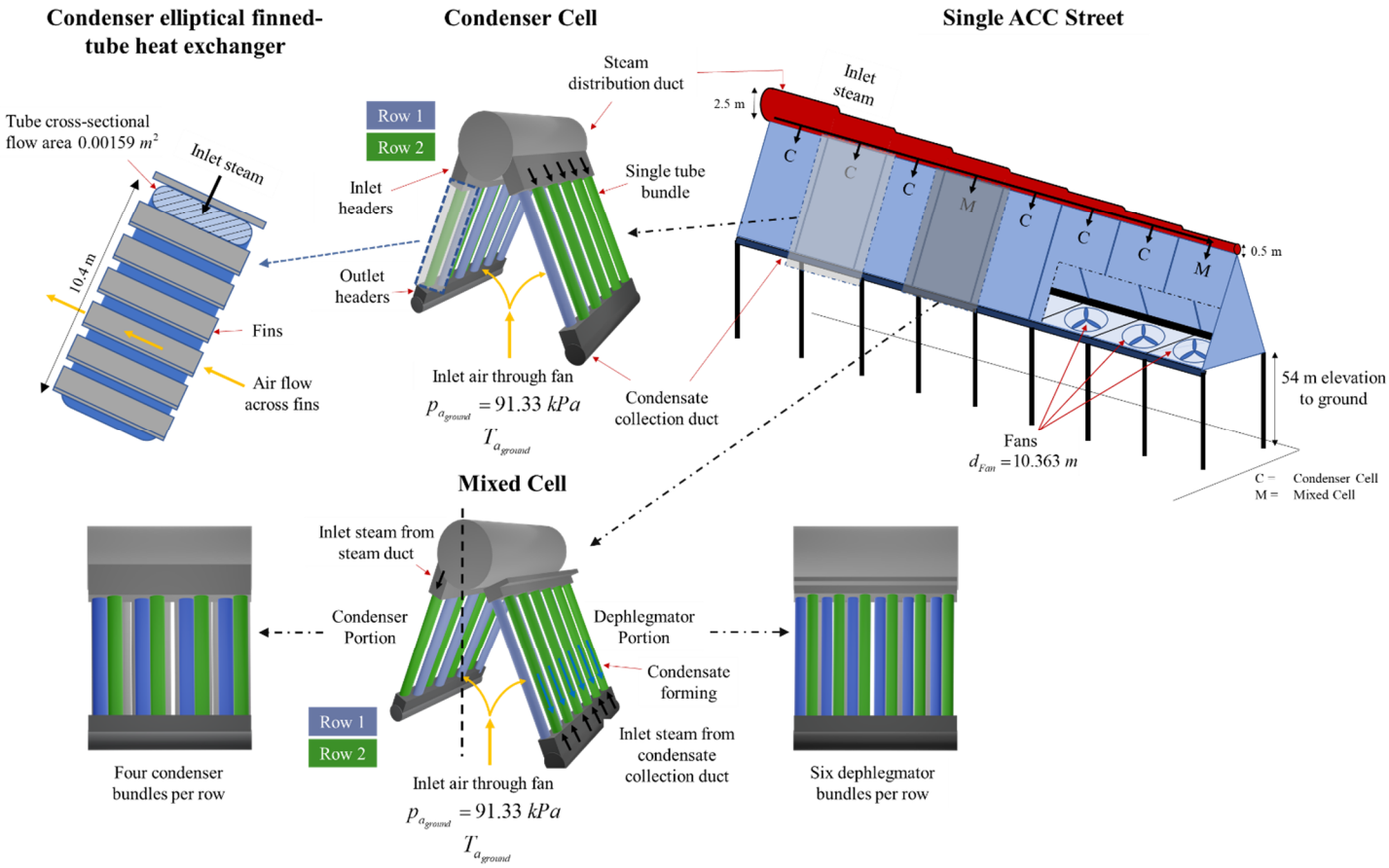


Figure 9: From left to right: Condenser elliptical finned-tube heat exchanger, a condenser cell, a mixed cell, and a single ACC street

Figure 9 shows a condenser heat exchanger, a condenser cell, a mixed cell, and a single ACC street. Each street had a total of six condenser cells and two mixed cells. Steam was distributed along a street to three condenser cells then a mixed cell followed by the last three condenser cells and the last mixed cell. The inlet ambient air pressure for all cells was also fixed at 91.33 kPa, with adjustable ambient air temperatures per cell.

All cells had an elevation of 54 m from the fan to ground level, with a fan diameter of 10.363 m and fan drive efficiency of 89.28%. A tube length of 10.4 m and 9.4 m was used for heat exchangers in a condenser cell and a dephlegmator respectively. Finned-tube heat exchangers with an elliptical cross-section were modelled as seen on the left of Figure 9. There were $n_{tb_1} = 52$ and $n_{tb_2} = 53$ tubes in a single tube bundle for rows 1 and 2 respectively for all cells. There were $n_b = 10$ bundles per row for both condenser and mixed cells and, therefore, 520 and 530 tubes in parallel

for rows 1 and 2 respectively. Mixed cells had six dephlegmator bundles and four condenser bundles as seen at the bottom of Figure 9. Each tube had an internal cross-sectional flow area of $0.00159\ m^2$ and circumference of $0.21341\ m$. The A-frame apex half-angle was specified as $\theta = 32°$. The total frontal air-side flow area for a single tube bundle, as shown in the middle of Figure 9, was $27.56\ m^2$.

The steam ducting into each street was modelled, as well as the ducting along a single street. The two main steam ducts had a diameter of $6.22\ m$. Each main steam duct provided steam to four streets. As seen on the right of Figure 9, the steam ducting diameter along a street linearly tapered off from $2.5\ m$ at the first cell in the street to approximately $0.75\ m$ at the last cell. The ducting was also simplified between the two main steam ducts and individual steam ducts for each street by assuming a single bend into each street. Figure 10 shows the overall ACC system layout with an indexing system.

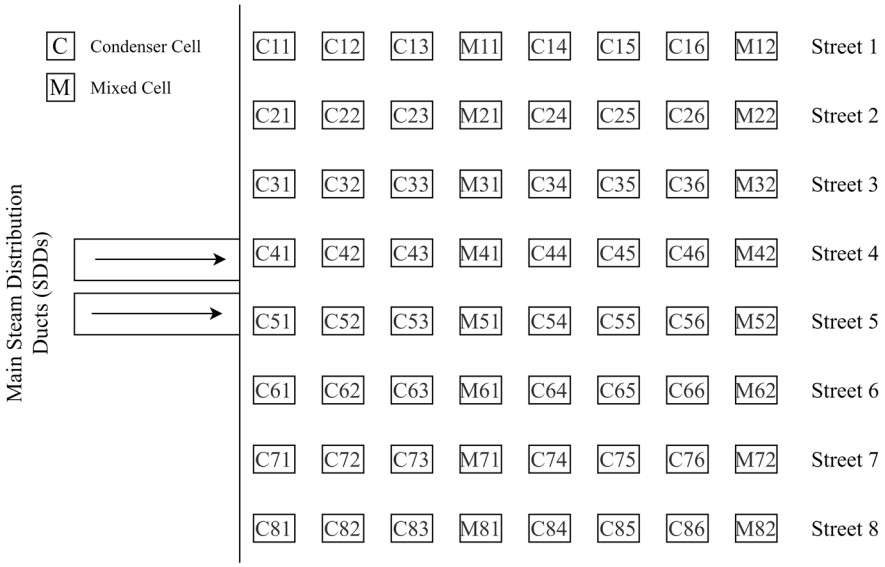


Figure 10: ACC system layout

The influence of varying ambient air temperatures and per cell air volume flow rates on the overall ACC system was investigated, including hot air recirculation effects and wind effects. To establish validation of the modelling methodology, model results were compared to steady-state site data under ideal conditions with no crosswinds or recirculation effects.

3.2 Governing Equations

The modelling methodology applied in the present work uses interconnected 1-D control volumes which solves for the mass, momentum and energy balance equations. The overall modelling methodology for each ACC cell consisted of a steam-side and an air-side control volume. For a given

cell, the steam-side consisted of steam-side flow through the heat exchanger tube bundles, while the air-side modelled the airflow path through the fan and the external surface area of the heat exchanger tube bundles. The steam-side control volumes such as the individual heat exchanger rows and steam conduits were connected to the air control volumes via heat transfer links as discussed below.

A 1-D modelling approach was applied using the thermofluid systems simulation software Flownex® SE 2019 [48]. A two-phase homogenous mixture assumption was used for steam flow, where the two-phases were distributed evenly over the flow area cross-section. For air-side flow, a single-phase fluid was modelled using the same equations, as below, but with $\overline{x}=1$. The vapour and condensate velocities and temperatures were assumed to be equal over the cross-sectional area of the flow element: $V_V = V_C = V_H$. Fluid properties were determined by weighted averages of vapour and condensate in the homogenous mixture. The average homogeneous volume fraction was given by:

$$\alpha_H = \frac{\rho_C \overline{x}}{\rho_C \overline{x} + \rho_V \left(1-\overline{x}\right)} \tag{1}$$

where $\overline{x}$ was given by:

$$\overline{x} = \dot{m}_V / \dot{m}_H \tag{2}$$

The mixture density, enthalpy and mass flow per unit area were calculated using:

$$\rho_H = \left(1-\alpha_H\right)\rho_C + \alpha_H \rho_V \tag{3}$$

$$H_H = \left(1-\overline{x}\right)H_C + \overline{x}\,H_V \tag{4}$$

$$\frac{\dot{m}_H}{A_H} = \rho_H V_H \tag{5}$$

Thermo-physical fluid properties such as conductivity and specific heats were calculated using the following relation:

$$\frac{1}{\zeta_H} = \frac{\overline{x}}{\zeta_V} + \frac{1-\overline{x}}{\zeta_C} \tag{6}$$

The homogenous mixture properties above were then used in the governing equations. Since a homogenous mixture assumption was used, only three governing equations were solved for the

working fluid, reducing computational expense and increasing numerical stability. Finite volume discretization was applied to derive a set of governing algebraic equations, developed by [49] and solved by Flownex® SE software. Pressures, densities and temperatures were defined at control volume centres, while velocities and volume flow rates were defined at control volume boundaries, seen in Figure 11. For transient simulation, a Crank-Nicholson fully implicit time step solver was used, with $\alpha = 1$.

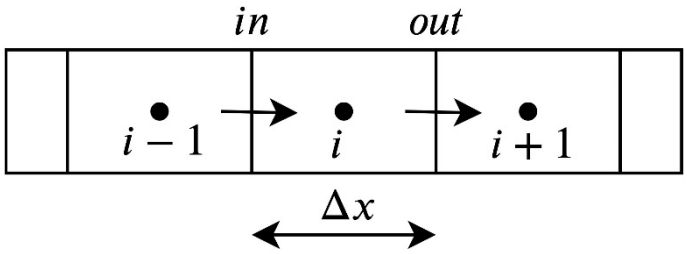


Figure 11: Discretized control volume

Mass conservation:

$$\cancel{V_i}\frac{(\rho_{H_i} - \rho_{H_i}{}^{o})}{\Delta t} + \rho_{H_{i+1}} Q_{out} - \rho_{H_{i-1}} Q_{in} = 0 \tag{7}$$

Momentum conservation:

$$\frac{\rho_{H_{i-1}} \Delta x}{A_i}\frac{(Q_{out} - Q_{out}{}^{o})}{\Delta t} + \frac{p_{i+1}}{p_{0_{i+1}}}\left(p_{0_{i+1}} - p_{0_i}\right) = -\left(\frac{f\Delta x}{D} + \sum K\right)\frac{\rho_{H_{i+1}} \left|V_{out}\right| V_{out}}{2} \tag{8}$$

where $\sum K$ is the sum of all secondary loss factors and $p_0 = p + 1/2\,\rho_H V_H{}^2 + \rho_H g z$.

Energy conservation:

$$\cancel{V_i}\left(\frac{\rho_{H_i} H_{0_i} - \rho_{H_i}{}^{o} H_{0_i}{}^{o}}{\Delta t}\right) - \cancel{V_i}\left(\frac{p_i - p_i{}^{o}}{\Delta t}\right) + \rho_{H_{i+1}} Q_{out} H_{0_{i+1}} - \rho_{H_{i-1}} Q_{in} H_{0_{i-1}} = \dot{Q}_i \tag{9}$$

where H_0 is defined as $H_0 = H_H + \frac{1}{2}V_H^2$.

An implicit pressure correction solution algorithm was used, solving the governing equations sequentially, as per [48], [49]. A detailed solver methodology can be observed further in [49].

3.3 Heat Transfer

The model accounted for internal film condensation, conduction in the heat exchanger tube walls, and air-side tube bank convection heat transfer. The overall heat transfer coefficient for a given heat exchanger row was found using:

$$UA_i = \frac{1}{\frac{1}{h_{C_i} A_{C_i}} + \frac{L_{cond_t}}{k_t A_{C_i}} + \frac{1}{h_{a_i} A_{a_i}}} \tag{10}$$

Kröger [5] developed empirical formulations for h_{C_i} and h_{a_i}, given as follows:

$$h_{C_i} = 0.9245 \left(\frac{L_t k_{C_i}{}^3 \rho_{C_i}{}^3 g \cos\left(90° - \theta\right) i_{fg_i}}{\mu_{C_i} m_{af} c_{pa_i} \left(T_{vm_i} - T_{ai_i}\right)} \right) \cdot \left(\frac{1}{\left[1 - e^{-UA_{C_i}/m_{af} c_{pa_i}}\right]} \right) \tag{11}$$

$$h_{a_i} = k_{a_i} \mathrm{Pr}_{a_i}{}^{0.333} \mathrm{Ny}_i \tag{12}$$

Ny_i was determined using:

$$Ny_i = b_{1_i} Ry^{b_{2_i}} \tag{13}$$

$$Ry_i = m_a / \left(\mu_{a_i} A_{fr} n_{tb_1} / n_{tb_2}\right) \tag{14}$$

Experimental values of b_{1_i} and b_{2_i} specific to the case study ACC system were used; however, typical values can be observed in [5].

Figure 12 shows the heat transfer resistances for the different segments of the model. The associated control volume was the entire length of a heat exchanger tube row as specified by Kröger [5]. Consequently, the heat transfer rate per row was calculated using the overall heat transfer coefficient.

$$\dot{Q}_i = UA_i \left(T_{vm_i} - T_{a_i}\right) \tag{15}$$

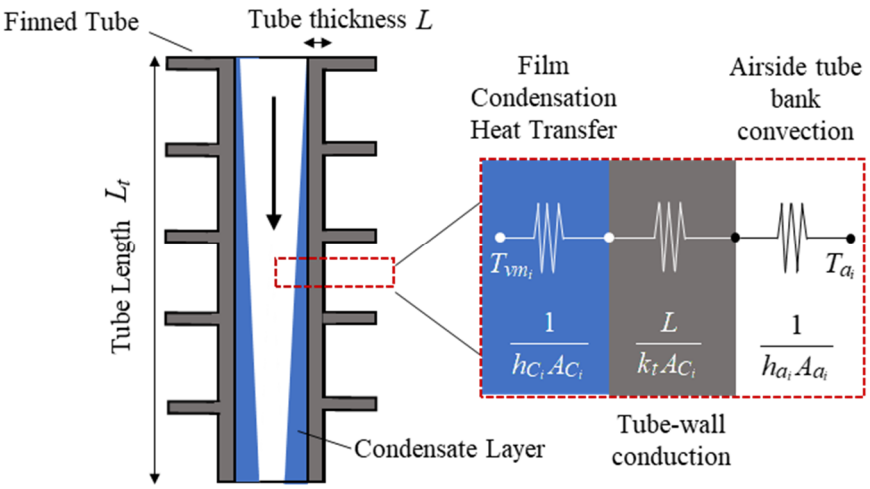


Figure 12: Heat transfer resistances for model segments

3.4 Steam-Side Modelling

3.4.1 Steam-side pressure drops

To capture steam-side pressure characteristics adequately, several pressure drops were incorporated from the SDDs to the outlet of the heat exchanger tube rows, derived by Kröger [5]. The pressure drop in the two main steam distribution ducts, Δp_{sd}, which was represented from the turbine outlet to before the steam ducting split into individual streets, was modelled using a secondary loss factor, $K_{sd} = 1.05$. The steam distribution ducting losses were modelled using a surface roughness of $200\,\mu m$ for the ducts supplying each street, specified for saturated steam ducts with minimal leakage [50]. The surface roughness acted as inputs to the Darcy-Weisbach equation, where the frictional pressure drop was calculated using [48]:

$$\Delta p_{duct} = \left(\frac{fL}{D}\right)\frac{\rho_H |V_H| V_H}{2} \tag{16}$$

$$f = \begin{cases} 0.25 / \left(\log\left[(e/3.7D) + 5.74/Re^{0.9}\right]\right)^2 & \text{if} \quad 5000 \le Re \le 10^8 \\ 64/Re & \text{if} \quad Re < 2300 \end{cases} \tag{17}$$

For $2300 < Re < 5000$, linear interpolation was used between the two expressions in Eq.(17). Additionally, a secondary loss factor of 0.3 was assumed for the bend into each street [50].

As steam entered the heat exchanger rows from the steam ducts, a contraction loss occurred for each row, Δp_{in_i}. The secondary loss factor was given by [5]:

$$K_{in} = \left(1 - \sigma^2 + K_c\right) \tag{18}$$

where σ was given by [5]:

$$\sigma = W_t L_t n_{tb_2} / A_{fr} \tag{19}$$

The frictional pressure drop in the heat exchanger row was given by [5]:

$$\Delta p_{fric_i} = \frac{0.1582 \mu_{V_i}{}^2 L_t}{\rho_{V_i} d_e{}^3 Re_{Vin_i}} \left(\frac{a_{1_i}}{2.75} \left(Re_{Vin_i}{}^{2.75} - Re_{Vout_i}{}^{2.75} \right) + \frac{a_{2_i}}{1.75} \left(Re_{Vin_i}{}^{1.75} - Re_{Vout_i}{}^{1.75} \right) \right) \tag{20}$$

where

$$Re_{Vn_i} = Re_{Vin_i} W_t / \left(2 L_t\right) \tag{21}$$

$$a_{1_i} = 1.0649 + 0.001041 Re_{Vn_i} - 2.011\text{E}-7 \cdot Re_{Vn_i}{}^3 \tag{22}$$

$$a_{2_i} = 290.1479 + 59.3153 Re_{Vn_i} + 1.5995\text{E}-2 \cdot Re_{Vn_i}{}^3 \tag{23}$$

Momentum recovery and gravitational pressure changes, Δp_{mom_i} and Δp_{grav_i}, were accounted for through the governing equations. An outlet expansion loss as the fluid stream exited the heat exchanger rows, Δp_{out_i}, was modelled through a secondary loss factor [5]:

$$K_{out} = \left(1 - \sigma_{34}\right)^2 - \left(1 - \sigma_{34}{}^2\right) \tag{24}$$

The frictional, momentum, gravitational, and outlet contraction pressure drop were grouped as a Δp_{HE_i} term.

3.4.2 Condenser steam-side flow path and discretization

To correctly capture the steam flow paths for a given condenser cell, multiple heat exchanger and flow conduit 1-D control volumes were used. Each condenser heat exchanger row was discretized as seen in Figure 13, with pressure drops at the respective components, including inlet headers, heat exchanger tube bundles, and outlet headers, presented earlier in Figure 9. Each condenser cell consisted of two heat exchanger rows. Condensate properties used in Eq. (11) were calculated at the outlet header.

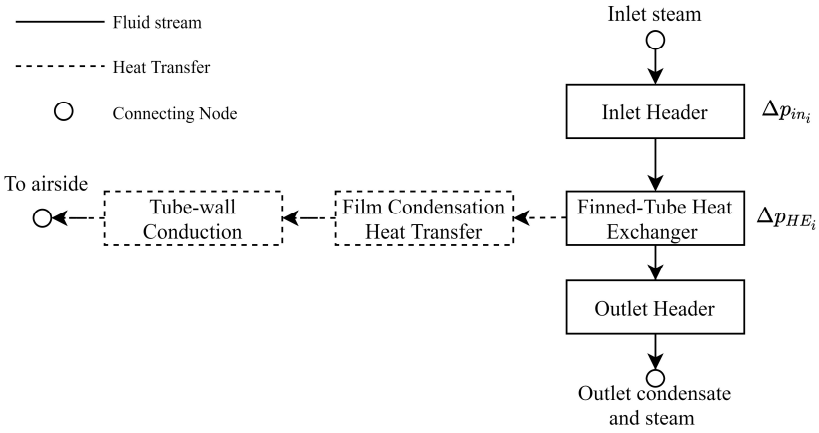


Figure 13: Condenser heat exchanger row discretization

3.4.3 Dephlegmator steam-side flow path and discretization

Dephlegmators were modelled differently to the condensers and received inlet saturated steam from the condensers. Rather than using fixed boundary conditions for the steam flow rate through the dephlegmator, the steam flow rate was based on the overall cooling capacity of the dephlegmator. This was done by combining the vapour flows in both heat exchanger rows into a single heat exchanger with a near-zero quality outflow boundary condition, seen in Figure 14. A near-zero quality boundary condition was assumed for numerical stability. A single control volume was modelled for both heat exchanger rows on the steam-side, however, the air-side still consisted of two separate control volumes for each row. This allowed for the heat rejection rates to be evaluated for each row.

To ensure an energy balance is enforced, the mass flow rate of saturated steam through the heat exchanger is changed until the energy equation is balanced. Once balanced, the overall cooling capacity of the dephlegmator for the given ambient boundary conditions would be known. The steam flow rate adjustment was made using successive minimization of the energy equation imbalance of all the dephlegmators. This method simplified and accounted for any backflow between rows.

The final dephlegmator discretization was used as explained above. However, another dephlegmator model was also developed. This additional dephlegmator model included a similar approach as used for condenser heat exchangers, where each heat exchanger row was modelled including a segment of row 1 to account for backflow. Detailed modelling and results can be observed in Appendix A.

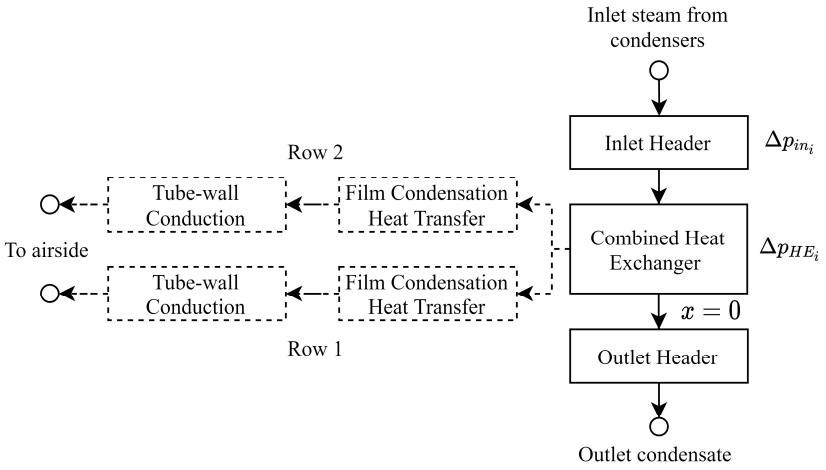


Figure 14: Dephlegmator combined heat exchanger row discretization

3.5 Air-side Modelling

A DALR of 0.00975 K/m was used to balance the pressure drops as air flowed past the fan assembly and heat exchangers and pressure rise through the axial fan with the change in air pressure with increased elevation. A pressure boundary condition was specified at the top of the A-frame calculated using Eq. (25).

$$p_{a_{BC}} = p_{a_{ground}} \left(1 - 0.00975\, H_{HEmid} / T_{a_{HEout}}\right)^{3.5} \left(1 - 0.00975 \left(H_{top} - H_{HEmid}\right) / \left(2T_{a_{ground}}\right)\right)^{3.5} \quad (25)$$

The air-side was also modelled using multiple interconnected 1-D control volumes. A series of flow conduit control volumes and secondary loss factors were used to model the air-side pressure drops, developed by Kröger [5]. These loss factors were incorporated through the governing equations, which calculated a specified air mass flow rate to satisfy the pressure boundary condition from Eq. (25), presented in Table 2. Additionally, the airside losses through the heat exchanger rows were included as $K_{\theta t}$, including the jetting, turning, and outlet header contraction losses. A detailed draft equation is presented in Eq.(26), where the ambient air pressure change on the left-hand side is used to balance the pressure drops on the air-side flow and the pressure rise from the axial fan. The mixed cell air-side discretization differed from the condenser cell due to the two distinct flow streams for the separate condenser and dephlegmator portions in the mixed cell.

$$p_{a_{ground}} - p_{a_{BC}} = \frac{K_{ts}}{2\rho_{a_{ground}}}\left[\frac{m_a}{A_{fr}}\right]^2 + \frac{K_{up}}{2\rho_{a_{FanInlet}}}\left[\frac{m_a}{A_e}\right]^2 - \frac{(K_{Fs}+K_{rec})}{2\rho_{a_{FanInlet}}}\left[\frac{m_a}{A_c}\right]^2 + \frac{K_{do}}{2\rho_{a_{FanOutlet}}}\left[\frac{m_a}{A_e}\right]^2 + \frac{K_{\theta t}}{2\rho_{HEavg}}\left[\frac{m_a}{A_{fr}}\right]^2 \quad (26)$$

Table 2: Air-side secondary loss factors [5]

Secondary Loss Factor	Value
K_{ts}	1.50
K_{up}	0.26
K_{rec}	0.31
K_{do}	0.40

Dry air properties were used for the model because of the DALR assumption. Fans were modelled using specified fan curves for volume flow rate and power [51], which provided a pressure rise to the air-side flow. Both condenser and mixed cells had individual fan curves modelled through polynomial fan equations and accounted for any scaling effects [51]. These fan curves allowed variable fan speeds to be incorporated into the model. A pressure recovery factor, K_{rec}, was selected through validation with site data. Initially, the fan motor powers between the site data and the model were matched by adjusting the fan speed. This resulted in an increased backpressure (due to a slightly lower fan speed). The recovery factor, K_{rec}, was then adjusted to account for the increase in backpressure, providing a further pressure rise at the fan. The recovery factor was used to account for the pressure recovery within the cell plenum as determined by [22] and also incorporated in [20], [23].

3.6 Condenser Cell and Mixed Cell Setup

Figure 15 shows the overall layout for a condenser cell. The heat transfer components interlinked the air-side and steam-side, with each “row” component represented by the heat exchanger shown in Figure 13. Air-side tube banks represented the overall air-side flow area of the heat exchangers. For a mixed cell, shown in Figure 16, four distinct flow streams were modelled, including the

condenser and dephlegmator air-side and steam-side. The condenser and dephlegmator air-side flow streams split into their respective flow paths downstream of the fan. “Combined rows” represented the combined heat exchanger in Figure 14.

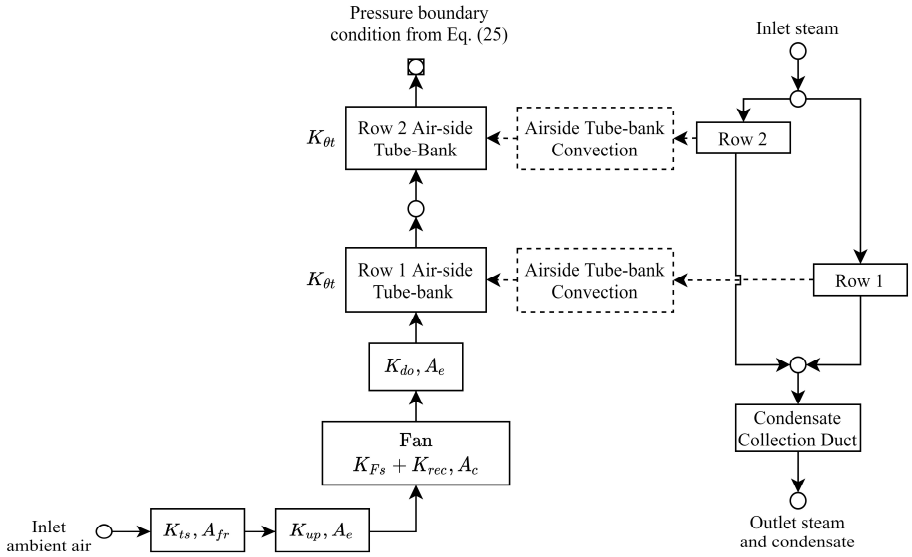


Figure 15: Condenser cell layout

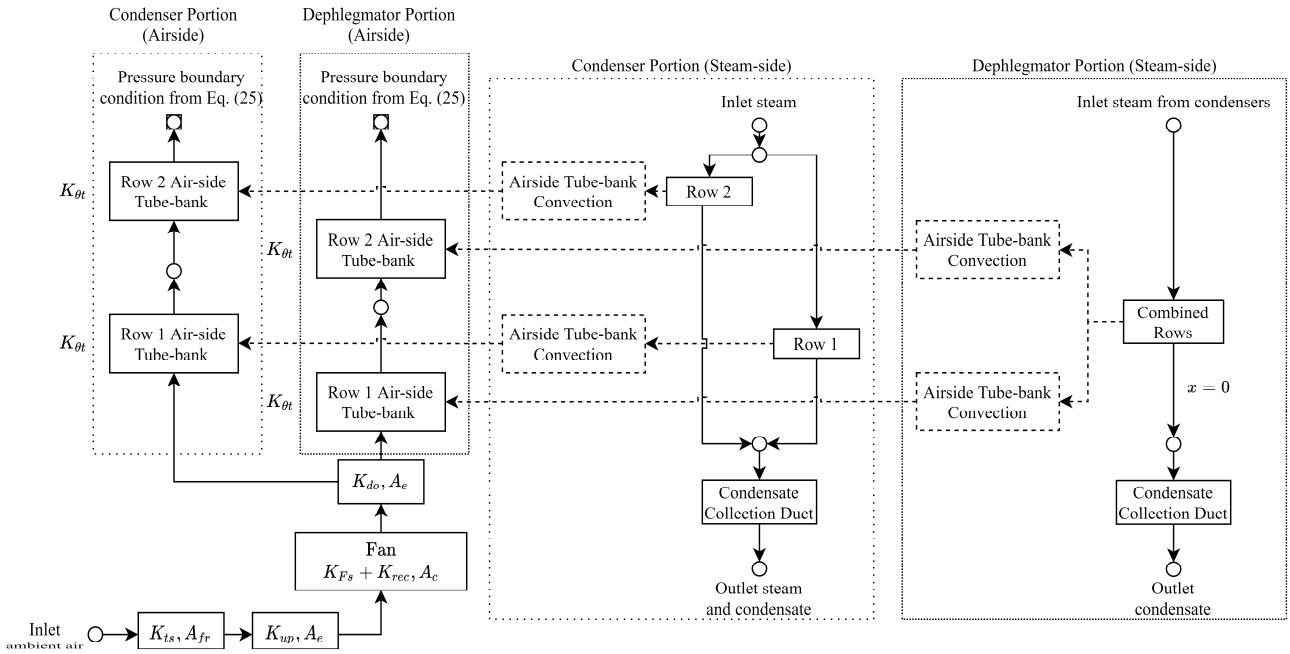


Figure 16: Mixed cell layout

Figure 17 (left) shows the condenser cell flow inputs, outputs, and associated boundary conditions. Inlet steam from the steam distribution duct entered the condensers. Figure 17 (right) shows the inputs, outputs, and boundary conditions for the mixed cell. The condenser portion of the mixed cell followed the same setup as a condenser cell. The dephlegmator portion received saturated steam ($\overline{x} = 1$) at the same pressure as steam exiting the condensers through a boundary condition. A mass flow rate boundary condition was used at the dephlegmator outlet to assign the steam mass flow rate that resulted in an energy balance. As a near-zero exit quality boundary condition was used, and an energy balance was achieved, the solution strategy would ensure that only condensate would flow out of the dephlegmator.

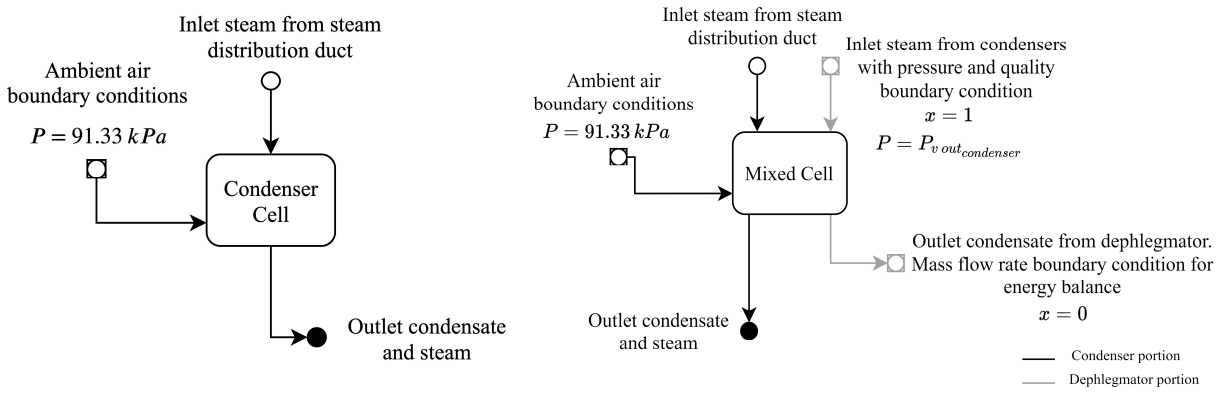


Figure 17: Condenser cell (left) and mixed cell (right) inputs, outputs, and boundary conditions

3.7 ACC System Setup and Solver Methodology

The case study ACC system model consisted of eight streets of cells, including six condenser cells and two mixed cells per street. Figure 9 (right) shows a single ACC street setup. Each cell had boundary conditions, as described in Figure 17. Figure 5 shows the case study ACC system setup, comprised of streets as shown on the right of Figure 9. A boundary condition was specified for inlet steam to the system, with a fixed quality. The steam pressure entering the system, or the backpressure, was an adjustable boundary condition which was used to ensure mass and energy balance of the total system.

Two main conditions had to be fulfilled to arrive at a steady-state solution. The first was ensuring an energy balance for the combined heat exchanger (Figure 14) for each dephlegmator. The second was that the summed condensate mass flow rate flowing out of all the dephlegmators and condensers should equal the total mass flow rate of steam entering the condenser system.

Two types of error feedback loops were used together in parallel while solving the system to a steady-state to ensure the above conditions were met. Each dephlegmator (totalling in 16 dephlegmators for the entire system) had a feedback loop monitoring the energy balance and adjusting the steam mass flow rate through the individual dephlegmators to maintain an energy balance. The primary feedback loop calculated the total condensate flowing out for all the dephlegmators and condensers, which was compared to the total steam flowing into the system. The total condensate outflow was calculated as the total liquid mass flow rate coming out of all the condensers and out of each dephlegmator. The backpressure would then be adjusted to ensure that the condensate outflow and the steam inflow was equal. If the total liquid mass flow rate was lower than the prescribed inlet steam mass flow rate, the backpressure would be increased. Likewise, if the total liquid mass flow rate was higher than the prescribed inlet steam mass flow rate, the backpressure would be decreased until a mass balance was achieved.

Both the error feedback loops above can potentially be executed in parallel. However, for numerical stability, the dephlegmator loop is solved followed by the backpressure loop. This would be repeated till convergence. This solving method limits the maximum change per iteration and proved to be more numerically stable. Consequently, this method was used to arrive at a converged backpressure.

Successive error minimization was used, which iteratively adjusted the backpressure and dephlegmator steam mass flow rates till the above conditions were met, indicating the system was at a steady-state. The solution methodology can be seen in Figure 18. Prior to this solution method, ambient air temperatures and fan speeds were specified for each cell. An initial guess for the backpressure and mass flow rates through each dephlegmator was also specified. As the backpressure was adjusted, all the dephlegmator feedback loops were continuously adjusted to maintain an energy balance.

Several relaxation parameters in Flownex® SE 2019 were adjusted for stability, seen in Table 3. The fractional convergence criteria for pressure was set to 1×10^{-4}, while the continuity and temperatures solvers were set to 1×10^{-6}.

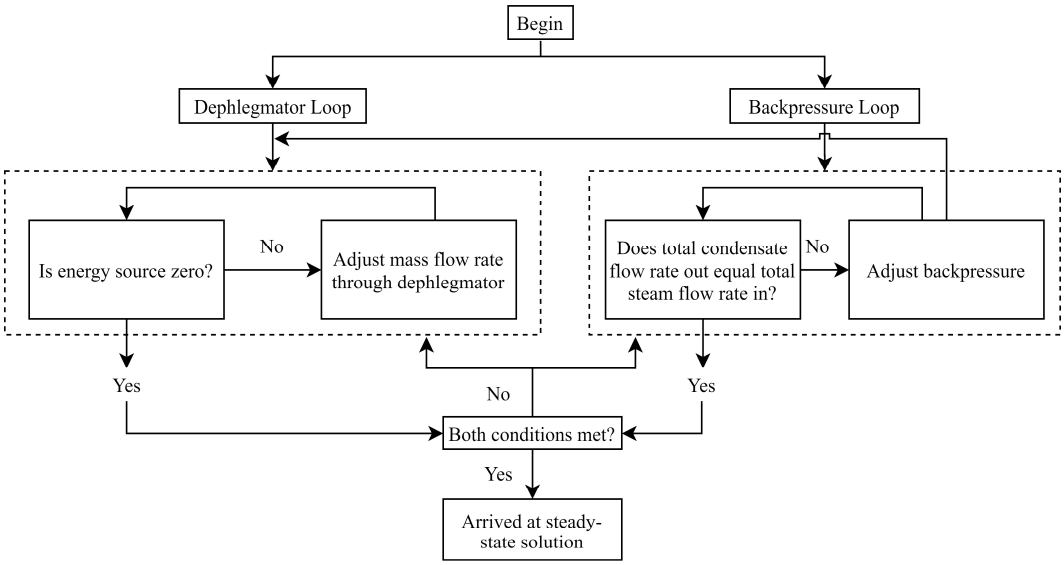


Figure 18: System solving methodology

Table 3: Adjusted relaxation parameters

Relaxation Parameter	Value
Pressure	0.3
Density	0.3
Mass	0.3
Temperature solver	0.3
Solid node temperatures	0.3
Two-phase density	0.3
Mass fraction	0.5
Negative pressure coefficient	0.5

4. ACC Thermofluid Modelling: Results and Discussion

This chapter was split into three sections, with the first section detailing validation of the model to site data as well as the existing lumped approach. The second section outlined the varying ambient air temperatures study, where a sensitivity analysis on several ACC performance parameters was conducted. The third and final section described the hot air recirculation and wind effects study.

4.1 Validation to site data

The model was validated with site data at an ambient air temperature of $27.21°C$ with good agreement in results. The developed model was also compared to the existing lumped approach based on Kröger [5]. The lumped approach differed from the developed model as it did not include the additional steam distribution ducting to each cell. The steam duct vapour distribution was only possible in the developed model as each cell was individually modelled, which allowed for the steam supply duct behaviour to cells within a street to be taken into account.

Table 4 shows the relative errors for model validation with site data, as well as the lumped approach results.

Table 4: Validation to site data at 27.21°C

Parameter	Site Data	Existing Lumped Approach	Model Results	Model Relative Error to Site Data (%)
Backpressure [*kPa*]	14.830	14.854	14.920	0.607
Total Heat Rejection Rate [*MW*]	928.754	930.696	927.900	0.092
Total Fan Power Consumption [*MW*]	12.141	12.161	12.159	0.155

A slight overestimate of backpressure was observed for the model compared to site data. Model results were agreeable within an acceptable range under 1%. The total fan power consumption of

the ACC system was matched through adjustment of fan speeds. Using the lumped model approach, along with the matched fan speeds, K_{rec} was adjusted until the lumped model's predicted backpressure was close to the backpressure from site data. A K_{rec} value of 0.255 was used for the lumped approach. Similarly, the K_{rec} value was adjusted to 0.310 for the developed model, in line with a recommended K_{rec} value of 0.3 specified by [22].

In comparison to the existing lumped approach, there were small differences in results attributed to the difference in flow properties and K_{rec} values for the volume flow rates. Additionally, the developed model had a slightly higher K_{rec} due to consideration of additional steam ducting distribution losses, not considered in the lumped approach. Slightly increasing K_{rec} resulted in an increased air volume flow rate, and consequently increased heat rejection rate, counteracting the additional vapour loss in the supply ducts.

4.2 Varying ambient air temperatures study

The model was then simulated for a range of operating ambient air temperatures from $25°C$ to $35°C$ at design conditions. Design conditions meant with no cross-winds, and uniform ambient conditions across all cells were specified. All condenser and mixed cells had the same inlet air temperature for the study. Consequently, the developed model and existing lumped approach displayed similar results. This study was mainly done for comparison between the developed model and lumped approach. For figures considering individual condenser cells and mixed cells in this study, results were taken from cells C11 and M11 respectively (refer to Figure 10). For system totals, results from all cells were summed together.

Figure 19 shows the backpressure and total heat rejection rate with varying ambient air temperature for the developed model and lumped approach. The backpressure rose steadily with increasing ambient air temperature as expected. For the model, an increase of $7.79\ kPa$ was observed over the $10°C$ temperature range. To quantify this change, for a given supercritical plant with a turbine efficiency of 95%, the overall loss in generation is $20.512\ MW$ from $25°C$ to $35°C$. This loss was observed at ideal wind conditions and no fan failures or recirculation effects, which would be a best-case scenario, not accounting for any additional ACC performance losses.

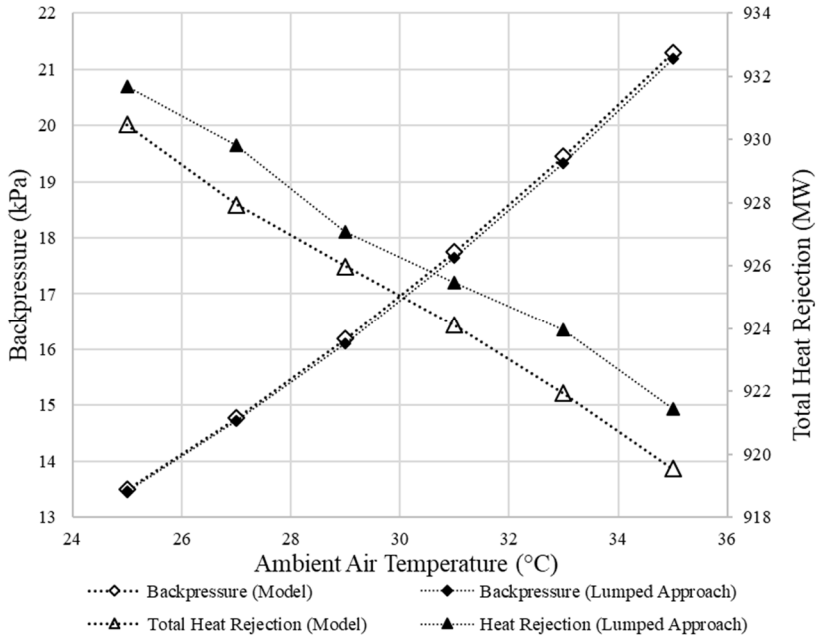


Figure 19: Backpressure and total heat rejection with varying ambient air temperatures

As expected, overall heat rejection rates decreased due to the increased air temperature. A loss in heat rejection rate of approximately 10.9 MW from 25°C to 35°C was observed for the model. The lumped approach produced larger heat rejection rates on average compared to the developed model. This was attributed to the consideration of the steam ducting losses incorporated into the model, which the lumped approach did not consider. The inlet steam to condensers in the developed model would be at a slightly lower vapour pressure and thus temperature, which resulted in slightly lower heat rejection rates.

Additionally, Table 5 shows the heat rejection rate map for street 1 at the two temperature extremities, 25°C to 35°C . Table 6 shows the condenser vapour inlet mass flow rates, steam duct pressure losses, and homogenous duct mass flow rate for individual cells in street 1 at 25°C and 35°C . The maps followed the same index and layout as specified in Figure 10, with each entry representing a cell. All maps presented followed a minimum (green) to a maximum (red) colour scale. For heat rejection rates, the mixed cells were split into condensers (C) and dephlegmators (D).

Table 5: Heat rejection rate map for street 1 at 25°C and 35°C (kW)

	C11	C12	C13	M11C	M11-D	C14	C15	C16	M12C	M12D
25°C	14934.5	14926.8	14917.3	5488.4	8015.0	14818.8	14883.0	14851.8	5460.3	8014.1
35°C	14685.2	14682.2	14678.6	5407.7	8060.2	14646.1	14666.0	14656.6	5397.9	8060.2

Table 6: Condenser vapour inlet mass flow rate, duct pressure loss, and duct mass flow rate for street 1 at 25°C and 35°C (kg/s)

		C11	C12	C13	M11C	C14	C15	C16	M12C
25°C	$\dot{m}_{V_{in}}$ (kg/s)	7.888	7.743	7.566	3.072	6.295	7.001	6.666	2.455
	Δp_{duct} (Pa)	17.07	20.86	25.39	30.31	55.33	67.61	67.50	22.69
	$\dot{m}_{duct_H}$ (kg/s)	52.75	44.20	35.81	27.61	24.28	16.69	9.48	2.65
35°C	$\dot{m}_{V_{in}}$ (kg/s)	7.838	7.700	7.532	3.056	6.383	7.030	6.720	2.482
	Δp_{duct} (Pa)	10.45	12.72	15.46	18.48	32.29	39.57	40.69	14.63
	$\dot{m}_{duct_H}$ (kg/s)	52.75	44.27	35.93	27.78	24.47	16.86	9.59	2.68

A reduction in heat rejection rate was observed for all cells across the temperature range, with condenser cells experiencing a more significant reduction in heat rejection rate compared to mixed cells. This reduction was because condenser cells received more vapour than the condensers in mixed cells, and consequently experienced a more considerable decrease in heat rejection rate from the increase in ambient air temperature. Condensers in condenser cells also had more tube bundles as compared to condensers in mixed cells. Condenser heat exchangers in mixed cells received less inlet vapour compared to condenser cells due to sizing differences between the two.

In Table 5, for condenser cells, heat rejection rates slightly decreased from the first cell to the sixth cell along the street, due to the inlet vapour flow rate observed in Table 6. The decrease in heat rejection rates along the street was driven by Δp_{duct}, which generally increased along the street. A sharp increase in Δp_{duct} was observed after M11, attributed to a smaller change in $\dot{m}_{duct_H}$ across M11C, which resulted in larger velocities due to the decreasing steam duct diameter along the street and a smaller change in $\dot{m}_{duct_H}$. Additionally, Δp_{duct} at $35°C$ decreased on average, due to the increased vapour density at higher vapour pressures resulting in lower vapour velocities, and in turn, a lower Δp_{duct}.

To further investigate the calculated steam-side pressure-drops in the heat exchangers, Table 7 shows total system results for the model and the existing lumped approach at $27°C$, at design conditions.

Table 7: Model results at 27°C ambient air temperature

Parameter	Existing Lumped Approach	Model Results
Backpressure [*kPa*]	14.755	14.774
Dephlegmator inlet vapour pressure [*kPa*]	13.695	13.846
Total heat rejection rate [*MW*]	930.696	927.932
Total fan motor power consumption [*MW*]	12.162	12.167
Air volume flow rate for condenser cell [m^3 / *s*]	658.145	667.482
Air volume flow rate for mixed cell [m^3 / *s*]	623.690	619.916

At uniform conditions, the model should produce similar results to the existing lumped approach, which was seen in Table 7. The slight variation in results, specifically with air volume flow rates, was due to the different K_{rec} factors used between the developed model and the existing lumped approach, as well as the steam ducting losses. However, recorded results were agreeable with less than a 1.5% relative error between the developed model and the existing lumped approach.

Figure 20 (left) shows the steam pressure drops for row 1 of condenser C11 at varying ambient air temperatures. Frictional, momentum, gravitational, and exit pressure changes were grouped as a total heat exchanger loss.

The pressure drops decreased with increasing ambient temperature, due to a reduction in secondary and frictional drops through the condenser at higher backpressures. The vapour density ratio for backpressures of 25 *kPa* and 15 *kPa* is approximately $\rho_{25[kPa]}/\rho_{15[kPa]} = 1.625$. Therefore, at higher backpressures due to increased ambient temperatures, the higher inlet vapour density results in lower vapour velocities through the condenser tubes, which reduces frictional and inlet/outlet pressure drops. The decrease in condenser pressure drop and heat rejection rate would result in the dephlegmator cells having higher pressure drops and heat rejection rates due to increased inlet vapour mass flow rates and pressures. Figure 20 (right) shows the same pressure drops for row 2 of a condenser. The total pressure drop was equal to row 1, as the flow stream combines as it exits the two rows, also preventing backflow from occurring. Inlet and total HE losses were similar and varied within only 2 *Pa* .

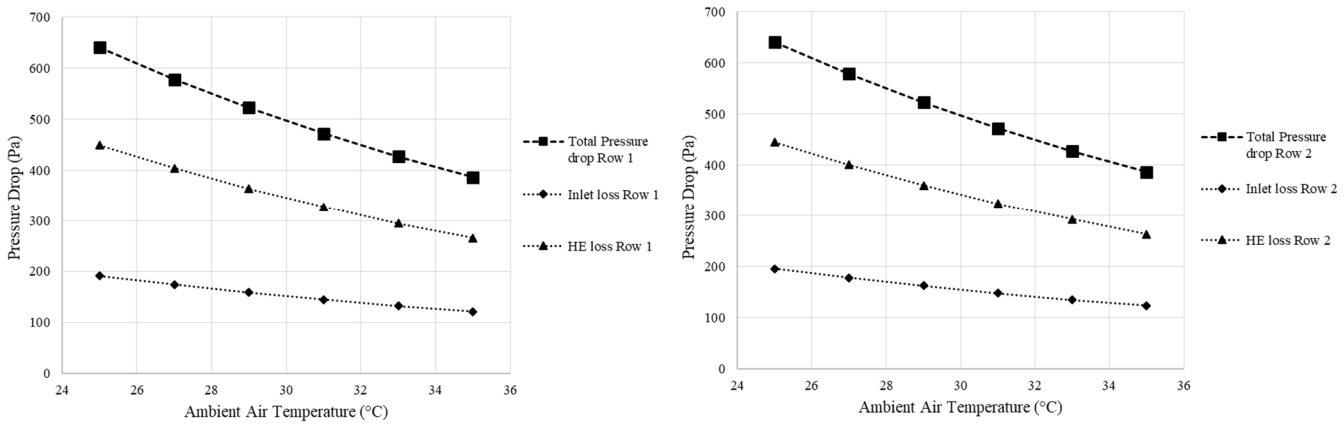


Figure 20: Row 1 (left) and row 2 (right) steam-side total pressure drops for condenser C11

Table 8: Vapour mass flow rates and pressure drops through each condenser heat exchanger row in condenser C11 at 27°C ambient air temperature

Parameter	Row 1	Row 2
Average inlet mass flow rate (kg/s)	4.241	4.288
Average outlet vapour mass flow rate (kg/s)	0.344	1.291
Δp_{in} (Pa)	174.068	177.969
Δp_{fric} (Pa)	537.654	498.156
$\Delta p_{mom+grav}$ (Pa)	-133.471	-97.878
Δp_{tot} (Pa)	578.25	578.25

Table 8 shows the average inlet and outlet vapour mass flow rates as well as pressure drops in condenser cell C11 at $27°C$. The total pressure drop across each row was equal to each other. Row 1 outlet vapour mass flow rates were significantly lower than row 2 due to the larger heat rejection rate in row 1, which received cooler inlet air than row 2.

In terms of pressure drop differences, Δp_{in} was slightly larger for row 2 due to the increased mass flow rate in row 2. The smaller Δp_{fric} in row 2 was attributed to the increased outlet vapour flow rate in row 2, which had an increased Re_{Vout_2} (Eq. (20)). In turn, there was a smaller difference between Re_{Vin_2} and Re_{Vout_2}, and consequently a lower Δp_{fric}. Row 1 had more momentum recovery due to the

lower vapour outlet flow rate compared to row 2. The increased Δp_{fric_1} and more momentum recovery in row 1 ensured a pressure balance between the two rows.

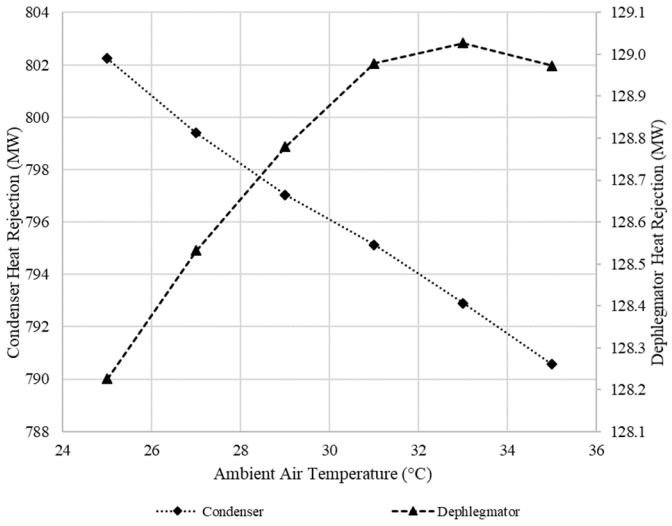


Figure 21: Total condenser and dephlegmator heat rejection rates with changing ambient air temperatures

Figure 21 shows the condenser and dephlegmator heat rejection rates with varying air temperatures. Condenser heat rejection rates decreased with increasing air temperatures, while dephlegmator heat rejection rates increased. This was explained by the additional vapour inlet to the dephlegmators that did not condense due to lower heat rejection rate in the condensers, in addition to higher dephlegmator inlet vapour pressures. Noticeably, dephlegmator heat rejection rates plateaued and started decreasing due to the increase in air temperature, counteracting the effect of additional inlet vapour mass flow rate and higher vapour pressure on heat rejection rates. The average total contribution of the dephlegmator heat rejection rates was 13.9% compared to the majority of heat rejection rates from condensers of 86.1%.

Figure 22 shows the heat rejection rates for the two rows in condenser cell C11 and the dephlegmator in M11 for all tube bundles in each row. Row 1 rejected more heat than row 2 due to the cooler inlet air, with a 2.2 *MW* average difference between the two condenser rows. For the condensers, heat rejection rates for both rows decreased with higher inlet air temperatures. However, heat rejection rates for row 1 decreased less compared to row 2, with an average decrease of 0.491% for row 1 compared to 3.24% for row 2 over the temperature range. This difference was attributed to the decrease in airside temperature difference over row 2. With higher inlet air temperatures, the inlet vapour pressure (backpressure) also increased. This vapour pressure increase helped counteract the effect of ambient air temperature increase on heat rejection rates

in row 1. However, for row 2, this effect is less pronounced as row 2 receives hotter air than row 1, resulting in larger heat rejection rate decreases than row 1.

For dephlegmators, the heat rejection rate for row 1 increased due to the increased inlet vapour pressure. Row 2 heat rejection rates decreased, also observed with the plateau and decrease in Figure 21. The decrease in row 2 heat rejection rates was due to the decrease in temperature difference between air and vapour with increasing ambient air temperatures, which was also compounded by the increasing heat rejection in row 1.

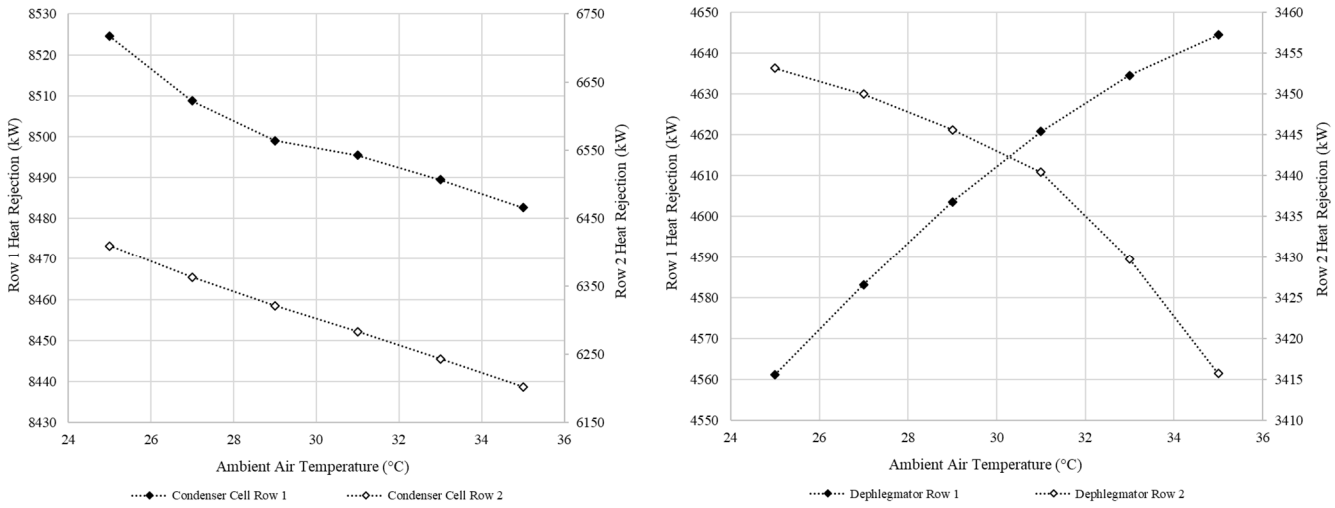


Figure 22: Heat rejection rates for condenser cell (C11) and dephlegmator (M11) rows with changing air

Heat transfer coefficients were also investigated across the two rows. Both condensate and air-side coefficients are shown in Figure 23 for a condenser in a condenser cell (C11) and a dephlegmator in a mixed cell (M11). For both condensate and air, row 2 had larger heat transfer coefficients than row 1 due to the specified heat transfer correlations used. Condensate coefficients increased with higher air temperatures, while air coefficients decreased. The increase in condensation heat transfer coefficients was due to higher vapour pressures (with higher ambient air temperatures) which resulted in increased thermal conductivity through the liquid film, which had an enhancing heat transfer effect.

Condensate coefficients were also much larger than the air coefficients. From Table 9, air-side convection thermal resistance was two orders of magnitude larger than both film condensation and heat exchanger tube wall conduction. Row 2 had a higher *UA* than row 1 due to different heat transfer correlations specified as well as the increased finned tube surface area (due to more tubes in row 2), counteracting the smaller temperature difference between steam and the hotter inlet air exiting from row 1.

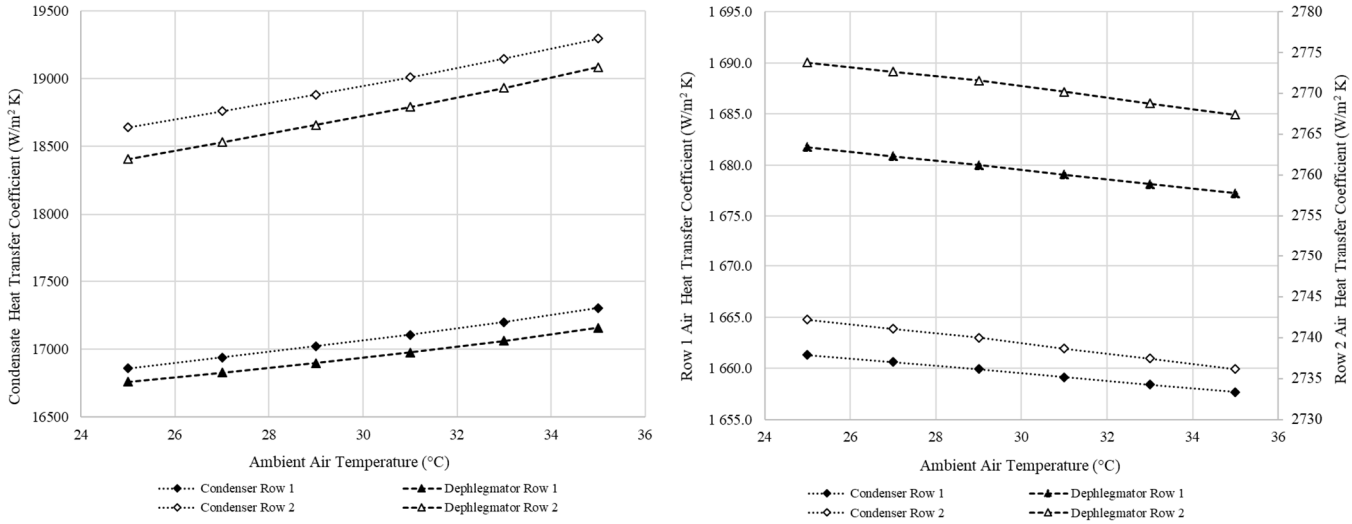


Figure 23: Condensate (left) and air (right) heat transfer coefficients for row 1 and 2 for condenser cell C11 and dephlegmator in M11 with changing ambient air temperatures

Table 9: Average heat transfer thermal resistances and overall heat transfer coefficient from 25°C to 35°C for condenser C11 and dephlegmators in M11

	Average Thermal Resistance [K/W]			
	Film Condensation	Heat Exchanger Tube Wall Conduction	Air-side Tube Bank Convection	Overall Heat Transfer Coefficient UA [W/K]
Condenser Row 1	5.076E-08	1.456E-08	2.228E-06	435957.1
Condenser Row 2	4.486E-08	1.429E-08	1.350E-06	709610.2
Dephlegmator Row 1	9.428E-08	2.685E-08	4.060E-06	239147.8
Dephlegmator Row 2	8.369E-08	2.634E-08	2.461E-06	388913.4

In addition to the heat rejection rates and heat transfer coefficients shown above, the air volume flow rate and fan power consumption for both condenser and mixed cells decreased linearly by 0.696% and 2.021% on average respectively. This was attributed to the density changes in air with increased temperatures, as well as the changes in various air-side losses with decreased heat rejection rates.

4.3 Hot air recirculation and wind effects study

The model was used to investigate off-design conditions, including hot air recirculation effects, as well as reduced air volume flow rate through a cell due to wind effects.

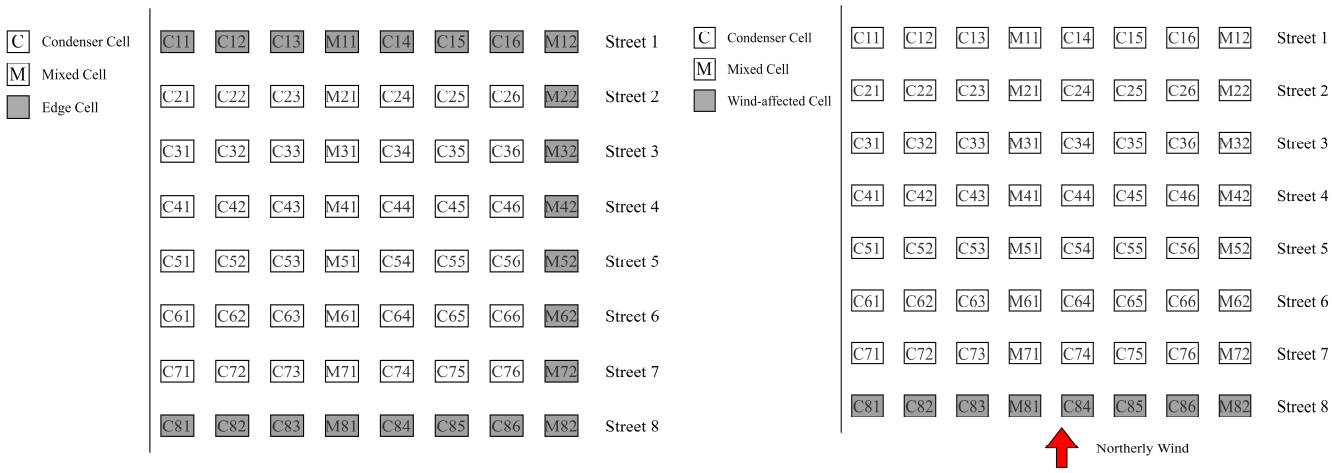


Figure 24: Edge cells modelled with recirculation (left) and wind-affected cells (right)

Recirculation effects occur when hot air at the outlet of the heat exchanger rows recirculates and re-enters the cell through the fan. Recirculation effects were modelled on edge cells of the system, as these cells would be affected most by any crosswinds impacting the ACC system. The affected cells included 18 condenser cells and four mixed cells. These were the two edge streets (streets 1 and 8) in addition to edge condenser cells for the remaining six streets, seen on the left of Figure 24. For the wind effect study, a northerly wind was modelled, impacting only street 8, seen on the right of Figure 24, which was an assumption for the case study. It is important to note that these case studies were mainly to demonstrate the model utility, and that actual operating cases of a real ACC system differed. These case studies serve as arbitrary conditions to demonstrate the model utility, based on possible off-design use cases of real ACC systems.

These off-design case studies were also implemented in the lumped approach through a cell-weighted average of affected and non-affected cells, where the affected parameter was averaged over the system. Inputs to the lumped approach could not be provided on a cell-by-cell basis, however, inputs to the developed model could be provided on a cell-by-cell basis due to the discretized modelling approach used. The only way to model off-design conditions in the lumped approach was through a cell-weighted value average over the ACC system. In the case of recirculation, this was the cell-weighted average of the recirculated air temperature and ambient air temperature. For wind effects, this was the cell-weighted average of fan speeds. For the developed model, a weighting of outlet air temperature to ambient air temperature was used. The

outlet air temperature was found at design conditions at $27°C$ ambient to be $47.25°C$. This outlet air temperature was then combined with the ambient air temperature for the recirculating cells, with up to a 60% outlet air temperature weighting. For the lumped approach, a cell-weighted average was used based on the recirculation air temperature and ambient air temperature. The cell-weighted average resulted in a single ambient air temperature over all cells, as the lumped approach only modelled a single condenser and mixed cell. Table 10 shows the recirculation air temperatures used for the developed model and lumped approach.

Table 10: Recirculation air temperatures (°C) for lumped approach and developed model

	Recirculation Weighting	0%	20%	40%	60%
Inlet air temperature ($°C$)	Lumped Approach	27.0	28.39	29.78	31.18
	Model Unaffected Cells	27.0	27.0	27.0	27.0
	Model Affected Cells	27.0	31.05	35.10	39.15

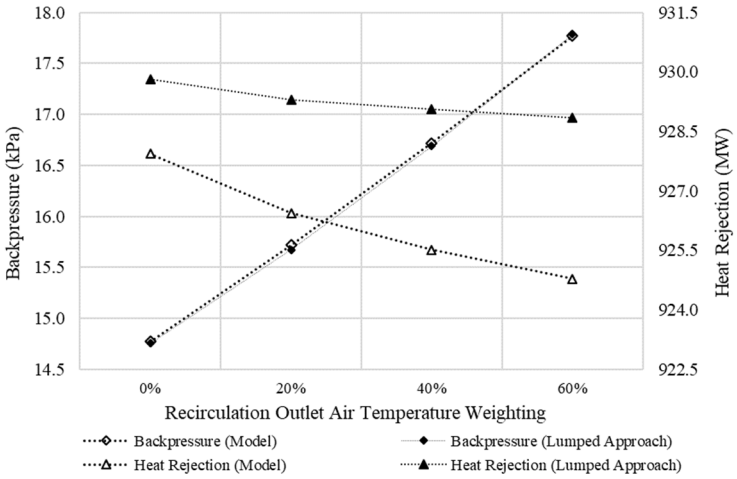


Figure 25: Recirculation effects on backpressure and total heat rejection at 27°C for model and lumped approach

Figure 25 shows the recirculation effects on backpressure and total heat rejection rate for the model and the lumped approach. An increase in backpressure was observed as recirculation increased. An increase of $3.00\ kPa$ and a decrease of $3.15\ MW$ from 0% to 60% recirculation was observed for the developed model. The lumped approach predicted nearly the same backpressure produced by the model and heat rejection rates slightly differed but under 0.5%. These similarities demonstrated

that modelling recirculation through a cell-weighted average using the lumped approach provided similar results as compared to modelling individual cells through the developed model.

Table 11: Heat rejection rate maps without (top) and with (bottom) recirculation effects at 27°C ambient temperature (MW)

	0% Recirculation Temperature Weighting (27°C)									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	14872	14866	14858	5468.8	8033.8	14779	14830	14806	5446.5	8033.5
2	14872	14866	14858	5468.8	8033.3	14779	14830	14806	5446.5	8033.1
3	14872	14866	14858	5468.8	8034.1	14779	14830	14806	5446.5	8032.8
4	14872	14866	14858	5468.8	8034.0	14779	14830	14806	5446.5	8033.3
5	14872	14866	14858	5468.8	8033.4	14779	14830	14806	5446.5	8033.1
6	14872	14866	14858	5468.8	8033.9	14779	14830	14806	5446.5	8033.7
7	14872	14866	14858	5468.8	8033.5	14779	14830	14806	5446.5	8033.2
8	14872	14866	14858	5468.8	8033.3	14779	14830	14806	5446.5	8033.5
	60% Recirculation Temperature Weighting (43.20°C)									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	10045	10043	10040	3694.3	5485.8	10011	10030	10021	3687.3	5485.0
2	17198	17193	17186	6332.3	9385.5	17152	17162	17146	3685.0	5485.1
3	17198	17193	17186	6332.3	9385.2	17152	17162	17146	3685.0	5485.5
4	17198	17193	17186	6332.3	9385.4	17152	17162	17146	3685.0	5485.9
5	17198	17193	17186	6332.3	9385.1	17152	17162	17146	3685.0	5485.6
6	17198	17193	17186	6332.3	9385.6	17152	17162	17146	3685.0	5485.6
7	17198	17193	17186	6332.3	9385.1	17152	17162	17146	3685.0	5485.3
8	10045	10043	10040	3694.3	5485.8	10011	10030	10021	3687.3	5484.8

Table 12: Outlet condensate mass flow rate maps without (top) and with (bottom) recirculation effects at 27°C ambient temperature (*kg / s*)

	0% Recirculation Temperature Weighting (27°C)									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
2	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
3	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
4	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
5	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
6	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
7	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
8	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
	60% Recirculation Temperature Weighting (43.20°C)									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	4.894	4.880	4.863	1.817	2.325	4.718	4.807	4.763	1.756	2.326
2	7.920	7.911	7.898	2.924	3.973	7.580	7.856	7.823	1.744	2.326
3	7.920	7.911	7.898	2.924	3.972	7.580	7.856	7.823	1.744	2.326
4	7.920	7.911	7.898	2.924	3.973	7.580	7.856	7.823	1.744	2.326
5	7.920	7.911	7.898	2.924	3.972	7.580	7.856	7.823	1.744	2.325
6	7.920	7.911	7.898	2.924	3.972	7.580	7.856	7.823	1.744	2.325
7	7.920	7.911	7.898	2.924	3.973	7.580	7.856	7.823	1.744	2.326
8	4.894	4.880	4.863	1.817	2.325	4.718	4.807	4.763	1.756	2.326

Table 11 and Table 12 show the heat rejection rates and condensate outlet mass flow rates for all cells without and with recirculation effects. Cells under recirculation effects exhibited lower heat rejection rates due to higher inlet air temperatures. Non-recirculation-affected cells had an increased heat rejection rate due to the increase in inlet vapour mass flow rates. The difference in recirculation and non-recirculation affected cells increased with higher recirculation temperatures, which contributed to the overall decrease in total heat rejection rates seen in Figure 25. As a result, the outlet condensate mass flow rates decreased for recirculation-affected cells, as seen in Table 12. A 29% decrease in outlet condensate flow rate was observed for C11 from 0% recirculation to 60% recirculation due to the decreased heat rejection rates for recirculation-affected cells. Moreover, a decrease of 28.75% and 31.17% in outlet condensate flow rate was observed for M11C and M11D from 0% recirculation to 60% recirculation. Non-recirculation-affected cells received

more inlet vapour, ensuring the total inlet vapour mass flow rate equalled the total liquid condensate flowing out of the ACC system. This is observed with an increase of $1.025\,kg/s$, $0.374\,kg/s$, and $0.595\,kg/s$ in outlet condensate for C21, M21C, and M21D respectively from 0% recirculation to 60% recirculation. With an increase in steam ducting loss along a street, inlet vapour pressures would decrease and consequently result in lower heat rejection rates, which was also observed with 60% recirculation. This resulted in cells at the beginning of the street receiving more vapour than cells at the rear of a street, which meant that condensate outlet mass flow rates were slightly lower for rear cells as compared to cells at the beginning of the street.

Wind effects were modelled through the incremented reduction of fan speeds, and in turn, air volume flow rates, across cells in street 8 on the right of Figure 24. This case considered a northerly wind impacting only street 8. For the lumped approach, a cell-weighted average of fan speeds was used, similar to the hot-air recirculation study, which was then used as the average fan speed for the system where all cells had the same fan speed. This approach was used to ensure the total air volume flow rate through the ACC for the two modelling approaches was similar. Table 13 shows the fan speeds used during the study for the lumped approach and developed model.

Table 13: Fan speeds (RPM) for lumped approach and developed model

	Fan Speed Reduction	0%	20%	40%	60%
Fan Speed (*RPM*)	Lumped Approach	103.9	100.7	98.1	95.6
	Model Unaffected Cells	103.9	103.9	103.9	103.9
	Model Affected Cells	103.9	83.12	62.34	41.56

Figure 26 shows the backpressure increasing non-linearly with higher fan speed reductions, as well as decreasing heat rejection rates for the model. For the lumped approach, there was a noticeable difference in backpressure and heat rejection rate trends compared to the present model. The backpressure rise was much lower as compared to the model predictions. The lumped approach also predicted an increasing heat rejection rate, which was the opposite predicted by the model, which had a decreasing heat rejection rate.

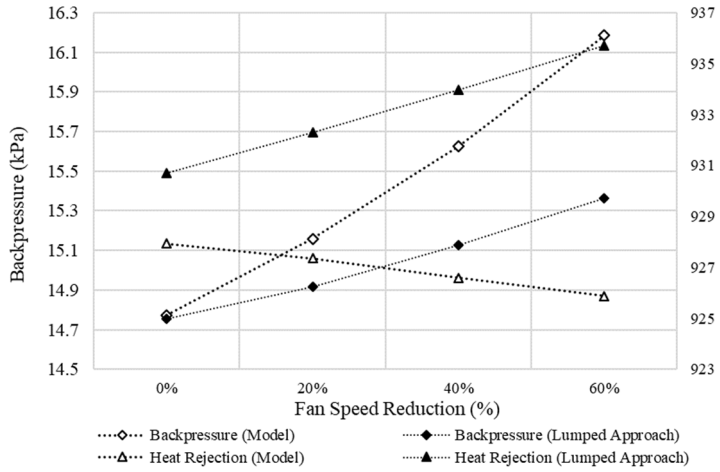


Figure 26: Reduced fan speed effects on backpressure and total heat rejection at 27°C for model and lumped approach

To better understand the heat rejection rate trends, heat rejection rates for row 1 and row 2 for the lumped approach were plotted against air mass flow rate for a condenser cell in Figure 27. Row 1 heat rejection rates increased with a decreasing air mass flow rate, while row 2 heat rejection rates decreased. Additionally, the backpressure increased with reduced air mass flow rates. This increased backpressure resulted in larger heat rejection rates for row 1 due to an increased vapour temperature in the heat exchanger tubes with a constant inlet ambient air temperature. The increased heat rejection rate, coupled with lower air mass flow rates, meant that the temperature difference in air temperature across row 2 was reduced and resulted in decreasing heat rejection rates. This was also observed in the developed model when implementing the same method used for the lumped approach through the cell-weighted average fan speeds.

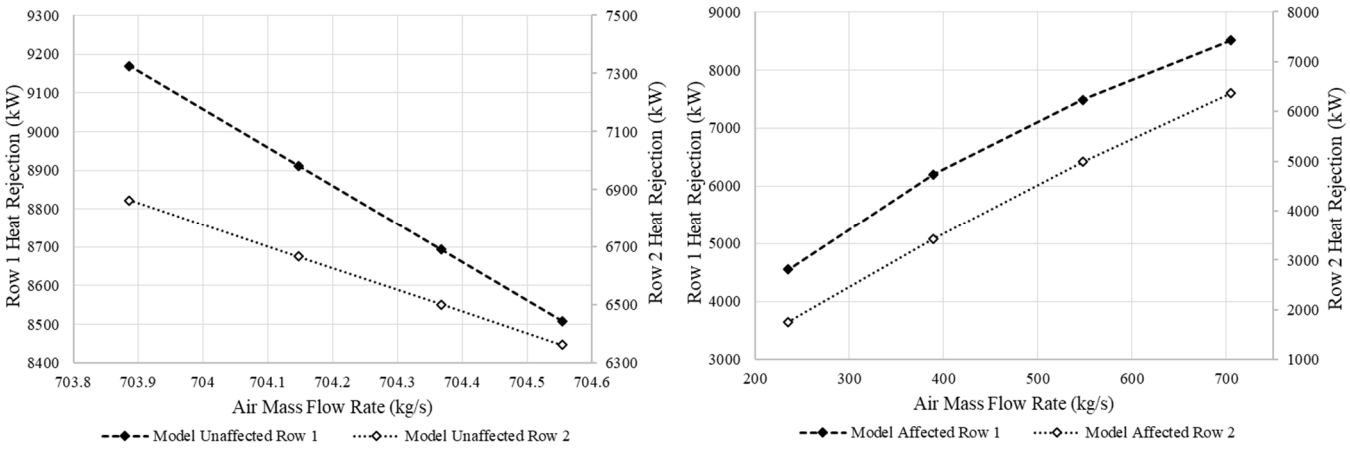


Figure 27: Lumped approach heat rejection rates for row 1 and row 2 of a condenser

Figure 28 shows the developed model predictions of heat rejection rates for row 1 and row 2 of C11 (unaffected by wind) and C81 (affected by wind) against air mass flow rates across the fan speed reduction range used for the study. For C11, heat rejection rates for both rows increased with decreased fan speeds. This was due to the increasing backpressure with lower fan speeds. As unaffected cells had fixed fan speeds, the air volume flow rates remained relatively unchanged. Consequently, with a higher backpressure the heat rejection rates increased for both rows. For C81, heat rejection rates decreased for both rows with decreasing fan speeds and thus lower air mass flow rates. The larger reduction in air mass flow rate counteracted the increased backpressure, which resulted in a decrease in heat rejection for affected cells. The presence of unaffected cells and affected cells in the developed model produced different predictions for the heat rejection as well as a more considerable increase in backpressure when compared to the cell-weighted average approach for the lumped approach.

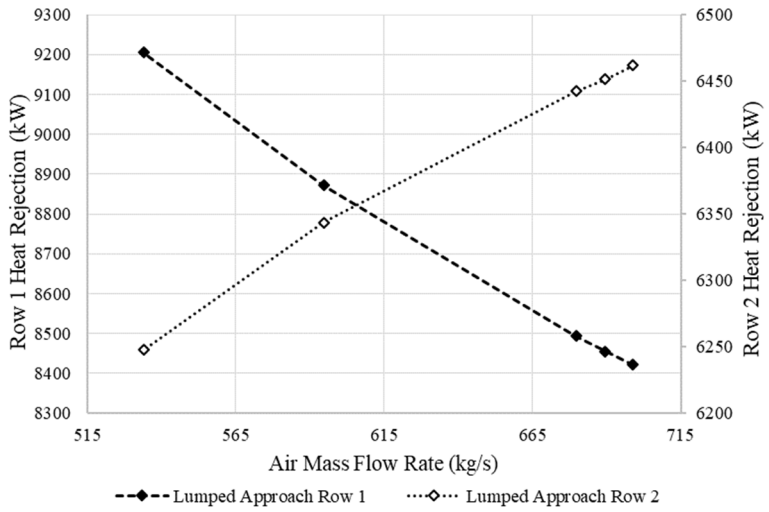


Figure 28: Model predictions for row 1 and row 2 heat rejection rates in unaffected C11 (left) and affected C81 (right)

Table 14: Heat rejection rate maps without (top) and with (bottom) fan speed reduction at 27°C ambient temperature (MW)

	0% Fan Speed Reduction									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	14872	14866	14858	5468.8	8033.8	14779	14830	14806	5446.5	8033.5
2	14872	14866	14858	5468.8	8033.3	14779	14830	14806	5446.5	8033.1
3	14872	14866	14858	5468.8	8034.1	14779	14830	14806	5446.5	8032.8
4	14872	14866	14858	5468.8	8034.0	14779	14830	14806	5446.5	8033.3
5	14872	14866	14858	5468.8	8033.4	14779	14830	14806	5446.5	8033.1
6	14872	14866	14858	5468.8	8033.9	14779	14830	14806	5446.5	8033.7
7	14872	14866	14858	5468.8	8033.5	14779	14830	14806	5446.5	8033.2
8	14872	14866	14858	5468.8	8033.3	14779	14830	14806	5446.5	8033.5
	60% Fan Speed Reduction									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	16032	16027	16020	5900.6	8712.9	15957	15993	15972	5879.5	8713.4
2	16032	16027	16020	5900.6	8713.3	15957	15993	15972	5879.5	8714.0
3	16032	16027	16020	5900.6	8713.4	15957	15993	15972	5879.5	8713.2
4	16032	16027	16020	5900.6	8713.6	15957	15993	15972	5879.5	8713.9
5	16032	16026	16019	5900.4	8712.9	15957	15993	15971	5879.3	8712.9
6	16032	16026	16019	5900.4	8712.9	15957	15993	15971	5879.3	8713.4
7	16032	16026	16019	5900.4	8712.9	15957	15993	15971	5879.3	8712.7
8	6314.4	6317.4	6320.6	2294.5	3446.6	6336.5	6329.1	6333.6	2302.1	3444.9

Table 14 shows cell heat rejection rates with and without fan speed reductions. The affected cells exhibited larger decreases in heat rejection rates compared to that observed in the recirculation study in Table 11. Additionally, row 2 heat rejection rates were significantly lower for cells with reduced air volume flow rates compared to cells with recirculation effects. Comparing the heat rejection rates per row for condenser cell C81 at 60% recirculation temperature weighting to C81 at 60% fan speed reduction, the heat rejection rates for row 1 and row 2 were $5.856\ MW$ and $4.221\ MW$ compared to $4.554\ MW$ and $1.760\ MW$ respectively. The significant reduction in row 2 heat rejection was due to the lower air volume flow rate, which resulted in air with higher inlet temperatures flowing into row 2.

Table 15: Outlet condensate mass flow rate maps without (top) and with (bottom) fan speed reduction at 27°C ambient temperature (kg/s)

	0% Fan Speed Reduction									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
2	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
3	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
4	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
5	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
6	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
7	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
8	6.895	6.880	6.863	2.550	3.378	6.730	6.807	6.770	2.491	3.378
	60% Fan Speed Reduction									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	7.379	7.369	7.357	2.727	3.672	7.247	7.313	7.278	2.678	3.672
2	7.379	7.369	7.357	2.727	3.672	7.247	7.313	7.278	2.678	3.672
3	7.379	7.369	7.357	2.727	3.672	7.247	7.313	7.278	2.678	3.672
4	7.379	7.369	7.357	2.727	3.672	7.247	7.313	7.278	2.678	3.672
5	7.378	7.368	7.356	2.726	3.672	7.241	7.302	7.267	2.673	3.672
6	7.378	7.368	7.356	2.720	3.672	7.241	7.302	7.267	2.673	3.672
7	7.378	7.368	7.356	2.720	3.672	7.241	7.302	7.267	2.673	3.672
8	3.416	3.394	3.369	1.256	1.454	3.164	3.288	3.226	1.171	1.453

Table 15 shows the change in condensate outlet vapour flow rate for cells with and without fan speed reductions, with a decrease in condensate outlet flow rate to cells with fan speed reduction due to lower heat rejection rates observed. For wind-affected cells C81, M81C, and M81D, a reduction of 50.46%, 50.75%, 56.96% was observed respectively. More condensate formed in non-wind-affected cells at 60% fan speed reduction compared to no fan speed reduction, with an increase of 0.484 kg/s (7.02%) for cell C11. This was due to the reduced condensate flowing out of wind-affected cells, which resulted in non-wind-affected cells condensing more steam to ensure that all the inlet steam to the ACC system was condensed. In comparison to the hot air recirculation study, a larger reduction in condensate outlet mass flow rate was observed for wind-affected cells compared to recirculation-affected cells. These differences demonstrated the effect of changing the ambient air temperature against changing the air volume flow rate for specific cells.

Table 16: Air volume flow rates without (top) and with (bottom) fan speed reduction at 27°C ambient temperature (m^3/s)

	0% Fan Speed Reduction									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
2	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
3	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
4	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
5	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
6	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
7	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
8	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
	60% Fan Speed Reduction									
	Cx1	Cx2	Cx3	Mx1C	Mx1D	Cx4	Cx5	Cx6	Mx2C	Mx2D
1	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
2	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
3	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
4	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
5	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
6	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
7	666.8	666.9	666.9	619.2	666.9	666.9	666.9	619.2	666.8	666.9
8	222.9	222.9	222.9	202.9	222.9	222.9	222.9	202.9	222.9	222.9

Table 16 shows the drastic change in air volume flow rate at the fan for cells with reduced fan speeds. The 67% decrease in air volume flow reduced both the condenser and mixed cells cooling effectiveness significantly. The reduction in heat rejection rate from lower air volume flow rates resulted in a more considerable backpressure spike to ensure all the inlet vapour condensed out of the system.

The model was limited with the 1-D approach used, as a uniform air volume flow rate was assumed throughout the fan inlet, which would not capture the detailed airflow profile. However, the developed model could quantify the effects of decreased air volume flow rate on system-wide performance. Further work is required to validate the off-design predictions with site data, or possibly a CFD model. A CFD model would allow for a precise mapping between wind speeds and the effects on the air volume flow rate to be accurately quantified.

5. Data-driven Surrogate Modelling

Chapter 5 contains all the relevant material for the data-driven surrogate modelling component of the project. The chapter was split into two main sub-sections; materials and methods, as well as results and discussion. With the thermofluid modelling covered in Chapters 3 and 4, this chapter pertains to the data generation, machine learning modelling, and web-application prototype (see Figure 6).

To utilise the developed 1-D thermofluid network model to generate data, slight simplifications were made to the original thermofluid network model to increase model stability. Model stability became particularly important when solving for a wide range of ambient and operating conditions to generate enough data points to train the machine learning models.

The detailed modelling of the steam distribution ducting to cells along an individual street (Section 3.4.1) was simplified using the lumped secondary loss factor, K_{sd}. The lumped steam duct secondary loss factor was increased from $K_{sd} = 1.05$ to $K_{sd} = 2.5$ in the simplified thermofluid network model. This simplification increased the stability of the model as the detailed steam-side flow distribution into individual cells along a street was no longer explicitly solved. Furthermore, the varying steam-side pressure drops in the supply ducts to each cell was ignored. Additionally, the K_{rec} value was adjusted to 0.263 for the simplified SDD model, compared to 0.310 used for the detailed SDD model. The increase in K_{rec} for the detailed SDD model compared to the simplified SDD model was required to overcome additional SDD pressure losses (showcasing the importance of detailed steam side modelling), by increasing the cooling airflow rates through the cells to correct for the lower steam-to-air temperature difference.

Figure 29 (left) shows the backpressure and total heat rejection rates at varying temperatures for both the detailed SDD model and simplified SDD model, while the graph on the right of Figure 29 shows the same accounting for wind effects or fan speed reduction. Both backpressure and total heat rejection rate predictions were relatively similar between the two models. The overall effect on ACC performance with varying ambient air temperatures and fan speed reduction was relatively similar when considering the detailed steam duct supply against a simplified steam distribution model. Heat rejection rates (with wind effects) were also similar, with relative differences under 0.1%.

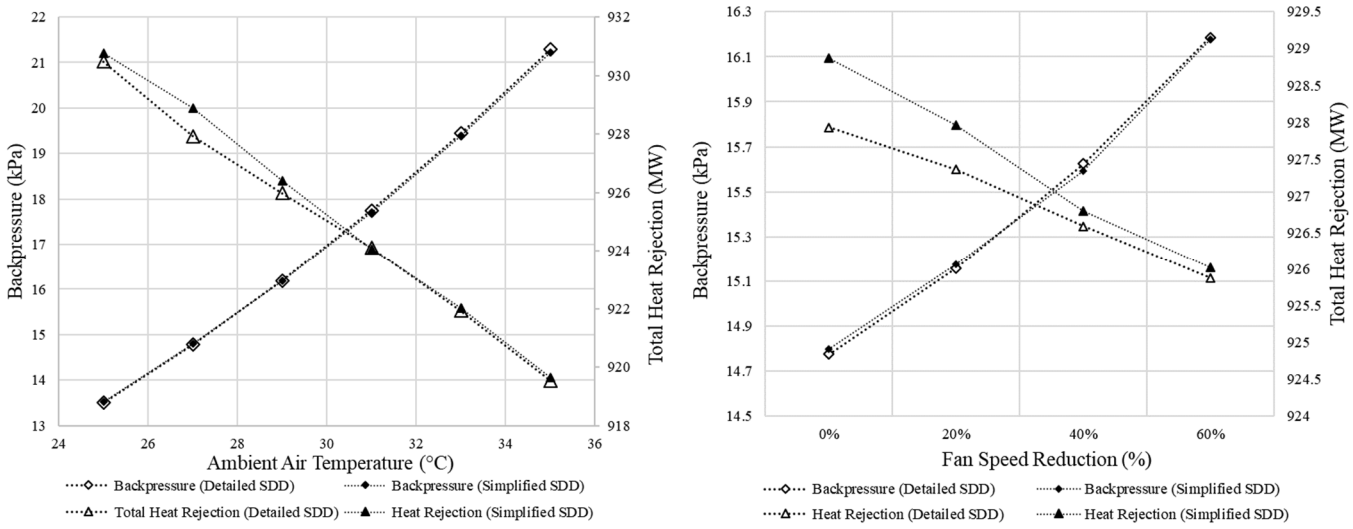


Figure 29: Backpressure and total heat rejection with varying ambient air temperatures (left) and fan speed reductions (right) for the detailed SDD and simplified SDD thermofluid network models

Table 17: Condenser vapour inlet mass flow rate for street 1 at 25°C and 35°C for the detailed and simplified SDD models (kg / s)

		C11	C12	C13	M11C	C14	C15	C16	M12C
Detailed SDD Model	25°C	7.888	7.743	7.566	3.072	6.295	7.001	6.666	2.455
	35°C	7.838	7.700	7.532	3.056	6.383	7.030	6.720	2.482
Simplified SDD Model	25°C	7.712	7.712	7.712	3.219	7.712	7.712	7.712	3.219
	35°C	7.715	7.715	7.715	3.211	7.715	7.715	7.715	3.211

Table 17 shows the vapour inlet mass flow rates to individual cells in street 1 at 25°*C* and 35°*C* for the detailed and simplified SDD models. The main difference between the two models was the constant vapour flow rate for condenser and mixed cells, respectively, at a given temperature for the simplified SDD model, in contrast with the detailed SDD model, which predicted a decreasing vapour flow rate to cells further along the street. This demonstrated the additional solving capability in the detailed SDD model, however, at a slight compromise for solver stability which was dependent on the initial guesses provided for the backpressure. The exclusion of the steam distribution ducting effects in the simplified SDD model increased stability considering a wide range of ambient conditions required for the data generation process.

Table 18 shows the heat rejection rate maps for street 1 at the two temperature extremities used in the study, 25°*C* and 35°*C*, for both detailed and simplified SDD models. The detailed SDD model displayed a gradient in heat rejection rates across the street for a given temperature, due to the

change in vapour mass flow rate distribution along the street. This differed compared to the simplified SDD model, where the heat rejection rate at a given temperature remained the same for condenser or mixed cells respectively along the street.

Table 18: Heat rejection rate map for street 1 at 25°C and 35°C for the detailed and simplified SDD models (kW)

		C11	C12	C13	M11C	M11-D	C14	C15	C16	M12C	M12D
Detailed SDD Model	25°C	14934.5	14926.8	14917.3	5488.4	8015.0	14818.8	14883.0	14851.8	5460.3	8014.1
	35°C	14685.2	14682.2	14678.6	5407.7	8060.2	14646.1	14666.0	14656.6	5397.9	8060.2
Simplified SDD Model	25°C	14861.3	14861.3	14861.3	5522.9	8006.2	14861.3	14861.3	14861.3	5522.9	8006.2
	35°C	14640.1	14640.1	14640.1	5446.1	8059.6	14640.1	14640.1	14640.1	5446.1	8059.6

The simplified SDD model displayed similar results in terms of overall backpressure and total heat rejection rates compared to the detailed SDD model. There were differences in terms of vapour distribution along cells within a street, and consequently, on heat rejection rates as well. However, the main aim of the data-driven surrogate model was to capture system-wide performance rather than performance on an individual cell level. Therefore, with the improvement in model stability and consideration of system-wide performance, the simplified SDD model was used for data generation.

5.1 Materials and Methods

The objective of data-driven surrogate modelling within the project was to leverage machine learning to increase the practicality and usability of the developed thermofluid model (simplified SDD model). The thermofluid model would be able to solve for given boundary conditions in under 30 minutes, which was relatively quick compared to CFD simulations. However, there was a need to decrease computation time to account for varying ambient conditions as well as possible unit load changes to be able to predict system performance in near real-time. A data-driven surrogate model was developed for a condition monitoring tool that could account for changes in these boundary conditions in shorter time increments. It is also important to consider that while the thermofluid network model was able to produce results on a cell-by-cell basis, the main aim of the data-driven surrogate model was to produce results on a system-wide basis. A cell-by-cell approach for a data-driven surrogate model would require vast amounts of generated data from the thermofluid network model, which was not practically feasible within the scope of the project. The developed condition monitoring tool through the data-driven surrogate model was used to predict system-wide performance, while the thermofluid network model had detailed cell-by-cell results.

5.1.1 Data generation

Data generation was a crucial part of developing a data-driven surrogate model. With a validated 1-D thermofluid network model available for generating performance data, a wide range of operating conditions could be considered for which to generate data. Consequently, limits were placed on the scope of operating conditions for this project due to time constraints. Three datasets were, therefore, considered with different input parameters for the datasets, shown in Table 19. A uniform inlet ambient air temperature was assumed for all cells in the ACC system to reduce the dimensionality of the input space.

Table 19: Input parameters for three datasets

Dataset 1	Dataset 2	Dataset 3
Ambient Air Temperature ($°C$)	Ambient Air Temperature ($°C$)	Ambient Air Temperature ($°C$)
ACC Inlet Steam Quality	ACC Inlet Steam Quality	ACC Inlet Steam Quality
ACC Inlet Steam Flow rate (kg / s)	ACC Inlet Steam Flow rate (kg / s)	ACC Inlet Steam Flow rate (kg / s)
	Number of Streets Switched Off (1, 2)	Wind Angle (Cardinal/Ordinal directions)
		Wind Speed (m / s)

Dataset 1 covered a wide range of ambient air temperatures, while dataset 2 considered lower temperatures, with the additional input consideration of switching off ACC streets. Dataset 3 included the effect of wind conditions on the system. All datasets had the ambient air temperature, ACC inlet steam quality, and ACC inlet steam flow rate as variable inputs.

Dataset 2 had an additional input where the number of ACC streets operating could be decreased by up to two streets. This additional input was required in consideration of lower ambient temperatures, particularly in winter, where the colder air increased the cooling capacity of the ACC system. Consequently, fewer cells would be required to completely condense inlet steam, as well as to maintain minimum backpressure to ensure forces on low-pressure turbine blades are controlled. Therefore, the consideration of switching off two streets accounted for an additional operating function of an ACC system. Due to scope limitation, data was generated for up to only two streets switched off; however, data for switching off more streets could be generated for future work. A minimum backpressure of $7\ kPa$ was used, and if a sample fell below this backpressure, it was removed from the result set.

Dataset 3 did not consider switching off ACC streets as an input variable but had two inputs for consideration of wind effects. These were the ambient wind speed as well as wind angle. Thus, dataset 1 had three input parameters, $k_1 = 3$, dataset 2 had $k_2 = 4$, and dataset 3 had $k_3 = 5$.

A design of experiments (DOE) was used to generate a range of data samples for each input parameter. Latin-Hypercube sampling (LHS) is a randomized sampling method developed by Mckay et al. [52] and was selected as the sampling method of choice. The sampling method selects n different samples from each of the k input parameters, $X_1 ... X_k$. Each parameter's range is divided into n non-overlapping intervals based on equal probability per interval. A single value from each interval is then selected at random based on the uniform probability density of the interval. This selection ensures repeated samples do not occur, and expands the range considered between multiple input parameters. Consequently, n values are obtained for each $X_1 ... X_k$. The n values obtained for X_1 are randomly paired with n values for X_2. This pair then combines with n values obtained up to X_k, forming a complete $k \times n$ input sample. Figure 30 [53] shows an example of LHS for two input parameters and a sample size of $n = 5$, where there are no overlapping samples between the intervals. Further detail on LHS can be found in [52], [54].

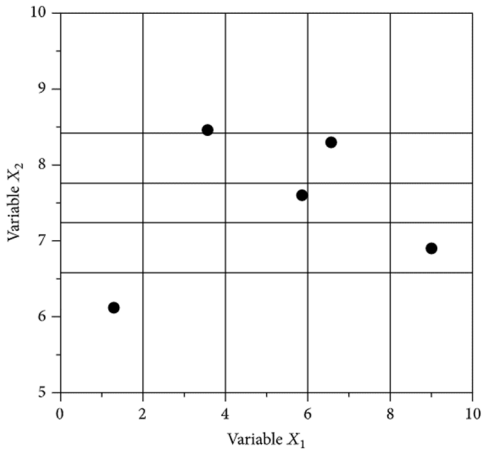


Figure 30: LHS example for two input parameters with a sample size of $n = 5$ [53]

LHS was incorporated using the existing Python library, pyDOE [55]. An n number of intervals was assigned for each input parameter and n samples generated for k input parameters. The samples of each input parameter ranged from 0 to 1 after LHS. A normal distribution was then used to transform the uniformly sampled points passed from LHS to a normally distributed set of inputs through a mean and standard deviation. An example is shown in Figure 31, with a normally distributed input space with $\mu = 0$ and $\sigma = 1$. The sampled inputs from LHS (y-axis) were transformed using a normal distribution (x-axis).

Normal distributions were therefore used for all input parameters shown in Table 19, with exceptions for discrete input parameters, which were the number of streets switched off and the wind direction. Additionally, the wind speed did not use a normal distribution, but an equally randomised sampling method, which is elaborated further on.

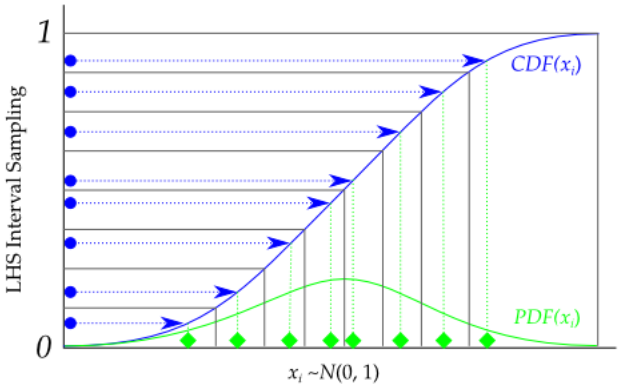
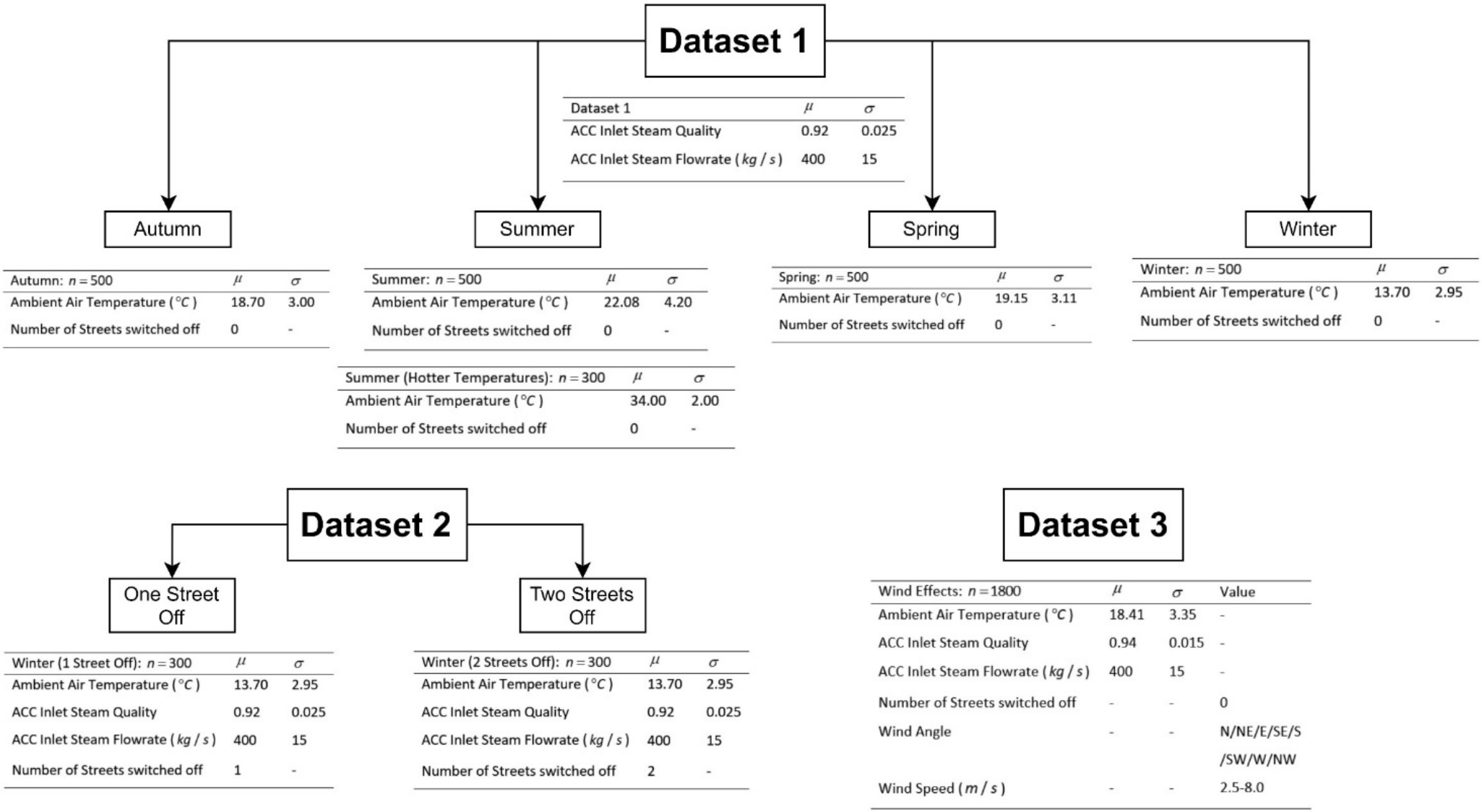


Figure 31: Transformation of input samples from LHS intervals (0-1) to a normal distribution $x_i \sim N(0,1)$ [55]

A further breakdown for each dataset is presented in Figure 32, along with the various means μ and standard deviations σ of normal distributions used for input parameters in each dataset. Dataset 1 was broken down into multiple DOEs. All of the DOEs in dataset 1 and 2 used the same μ and σ for the ACC inlet steam quality and flow rate. The inlet steam quality had a relatively small σ, based on a constraint on at the turbine outlet to prevent droplet formation. The inlet steam flow rate had a $\mu = 400\ kg/s$ based on the unit capacity at full load, with a $\sigma = 15\ kg/s$. The σ was again relatively small, with the focus on full unit loading, and therefore smaller variance in inlet steam flow rate. Increasing the σ would increase the size of the input space used and would, therefore, require more input samples to avoid a sparse dataset.

Dataset 1 was broken down into seasonal allocations based on the ambient air temperature input parameter ranges which corresponded to temperatures experienced during the different seasons. Historical weather data was used to develop the μ and σ on a seasonal basis [56]. It is important to note that a seasonal approach was used to ensure coverage of the ambient air temperatures, rather than using the season as an input parameter for predictions. Consequently, DOEs for autumn, summer, spring, and winter, were created with their respective μ and σ as presented in Figure 32. These DOEs were developed for all ACC streets operating (0 streets switched off). Each of these DOEs had $n = 500$ samples. An additional DOE was developed for summer, which considered higher temperatures not captured in the initial summer DOE, with $n = 300$. Therefore, for dataset 1 where all ACC streets were operating, 2300 input samples were generated.



Dataset 1	μ	σ
ACC Inlet Steam Quality	0.92	0.025
ACC Inlet Steam Flowrate (kg/s)	400	15

Autumn: $n = 500$	μ	σ
Ambient Air Temperature ($°C$)	18.70	3.00
Number of Streets switched off	0	-

Summer: $n = 500$	μ	σ
Ambient Air Temperature ($°C$)	22.08	4.20
Number of Streets switched off	0	-

Summer (Hotter Temperatures): $n = 300$	μ	σ
Ambient Air Temperature ($°C$)	34.00	2.00
Number of Streets switched off	0	-

Spring: $n = 500$	μ	σ
Ambient Air Temperature ($°C$)	19.15	3.11
Number of Streets switched off	0	-

Winter: $n = 500$	μ	σ
Ambient Air Temperature ($°C$)	13.70	2.95
Number of Streets switched off	0	-

Winter (1 Street Off): $n = 300$	μ	σ
Ambient Air Temperature ($°C$)	13.70	2.95
ACC Inlet Steam Quality	0.92	0.025
ACC Inlet Steam Flowrate (kg/s)	400	15
Number of Streets switched off	1	-

Winter (2 Streets Off): $n = 300$	μ	σ
Ambient Air Temperature ($°C$)	13.70	2.95
ACC Inlet Steam Quality	0.92	0.025
ACC Inlet Steam Flowrate (kg/s)	400	15
Number of Streets switched off	2	-

Wind Effects: $n = 1800$	μ	σ	Value
Ambient Air Temperature ($°C$)	18.41	3.35	-
ACC Inlet Steam Quality	0.94	0.015	-
ACC Inlet Steam Flowrate (kg/s)	400	15	-
Number of Streets switched off	-	-	0
Wind Angle	-	-	N/NE/E/SE/S/SW/W/NW
Wind Speed (m/s)	-	-	2.5-8.0

Figure 32: Breakdown of developed datasets

Dataset 2 was broken down into two segments with a single DOE, with one or two streets switched off. The ideology presented earlier behind switching off ACC streets pertained to lower temperatures in winter. Accordingly, the ambient air temperature μ and σ used for dataset 2 was the same as the winter DOE for dataset 1.

Dataset 3 had a single DOE configuration. The ambient air temperature range used covered a 95% confidence interval of the combined seasonal normal distributions used in dataset 1. This assumption was deemed sufficient to cover a wide range of ambient air temperatures for the local region while minimising the number of generated samples. Additionally, the inlet steam quality was adjusted, with an increased μ and decreased σ compared to datasets 1 and 2. These changes were done to narrow the distribution of steam quality around 0.94 since the inlet steam quality fluctuated less than other parameters, such as the inlet steam flow rate. As previously mentioned, for dataset 3, all ACC streets were operating (0 streets switched off). To capture wind effects in the DOE, both the wind angle and wind speed used random sampling rather LHS.

The thermofluid network model accounted for wind effects previously through a reduction in air volume flow rate (by reducing fan speeds for affected edge cells), seen in Section 4.3. This was used to demonstrate the utility of the model and the effect of cross-winds on performance. For a condition monitoring tool, the use of wind speed as an input parameter was more practical as opposed to air volume flow rate reduction. A reduction in air volume flow rate would be dependent on the wind speed and resulting cross-winds. In the data generation process, the wind angle parameter indicated which cells were affected by wind, while the wind speed parameter indicated the air volume flow rate reduction that the affected cells exhibited.

Due to a lack of available site data with wind speeds, a strong link between wind speeds and a reduction in air volume flow rates for wind-affected cells could not be made. A CFD model of the utility-scale ACC system would be required to investigate the relationship between wind speeds and the effects on the air volume flow rate through the cells. However, there was no existing CFD model for the ACC system considered in this work at the time of publication. As a last resort, data from Engelbrecht [20] for the air volume flow rate reduction resulting from cross-winds was used to develop a relationship for the present work. It is important to note that the developed CFD model from Engelbrecht [20], was based on a different utility-scale ACC system which contributed further to the uncertainty in the wind speed and air volume flow rate reduction relationship.

Nonetheless, the following process was used to develop a link between wind speeds and fan volume flow rate reduction for the present work. Figure 33 shows the total air volume flow rate for a 30 cell ACC system with increasing wind speeds simulated by Engelbrecht [20]. A cross-wind in the $y-$ direction, as shown in Figure 33, was simulated for the A-fan, B2a-fan, and combined ACC. The A-

fan ACC was used for reference in the present work as simulated A-fan results were closest in comparison to the air volume flow rates observed in the current work.

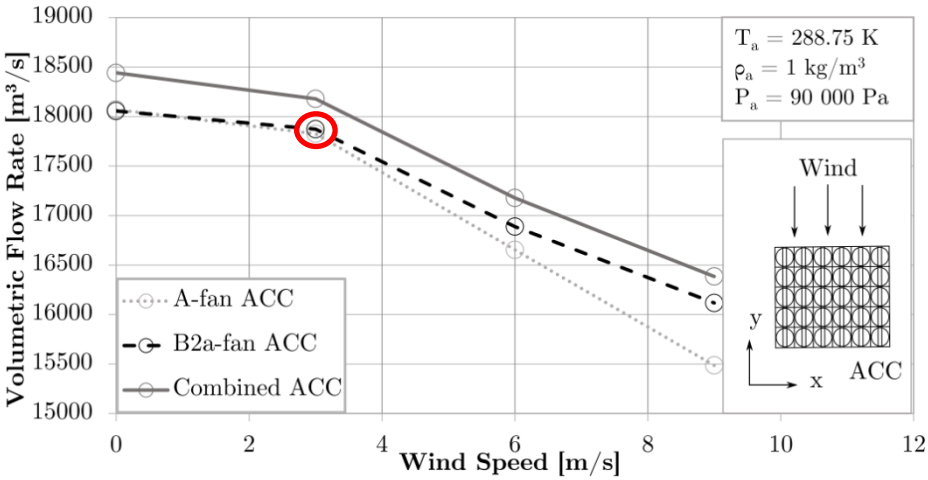


Figure 33: Total air volumetric flow rate subject to y-direction cross-wind by Engelbrecht [20]

The total air volume flow rate was divided across 30 cells to find the average air volumetric flow rate per cell. The average inlet air velocity to the ACC cell predicted for the thermofluid network model was approximately 2.8 m/s at design conditions, taken as $\dot{Q}_a / A_{air}$. Consequently, this data point was used as a baseline. The percentage in air volume flow rate reduction between the highlighted data point (red) and the next two data points in Figure 33 was approximately 6.58% and 10.17% respectively. This percentage reduction in air volume flow rate was then applied to the average air volume flow rate predicted by the thermofluid network model at the same ambient conditions ($15.6\ °C$). Table 20 shows the air volume flow rate reductions found for Engelbrecht [20] based on the wind-speed and applied to the air volume flow rate predicted by the thermofluid model in the present work.

Table 20: Air volume flow rate reduction based on wind speed for Engelbrecht [20] and present work

	Average Air Volume Flow Rate (m^3/s)		
Wind Speed (m/s)	Engelbrecht [20]	Present Work	Fan Speed Reduction (%)
3.0	603.01 (base)	668.95 (base)	0.0
6.0	563.33 (-6.58%)	624.94 (-6.58%)	46.64
9.0	541.77 (-10.17%)	600.91 (-10.17%)	75.84

Table 21: Air volume flow rates for a condenser cell with increasing fan speed reductions predicted by thermofluid network model at 15.6 °C with a westerly wind direction

Fan Speed Reduction (%)	0	10	20	30	40	50	60	70	80
Average Air Volume Flow Rate (m^3 / s)	669.0	659.7	650.1	640.3	631.0	621.9	612.4	604.2	598.5

Using a westerly wind direction (corresponding to the y-direction wind simulated by Engelbrecht [20]), the average air volume flow rate for condenser cells was found for increasing fan speed reductions using the thermofluid network model in Table 21. These results were then used to interpolate the air volume flow rates for the present work presented in Table 20. Consequently, the fan speed reduction column in Table 20 was produced, which established a relationship between wind speed and fan speed reduction. Figure 34 shows the three plotted data points from Table 20 for the fan speed reduction with varying wind speeds. Subsequently, a second-order polynomial trendline was fitted, allowing for predictions of fan speed reduction for respective wind speed.

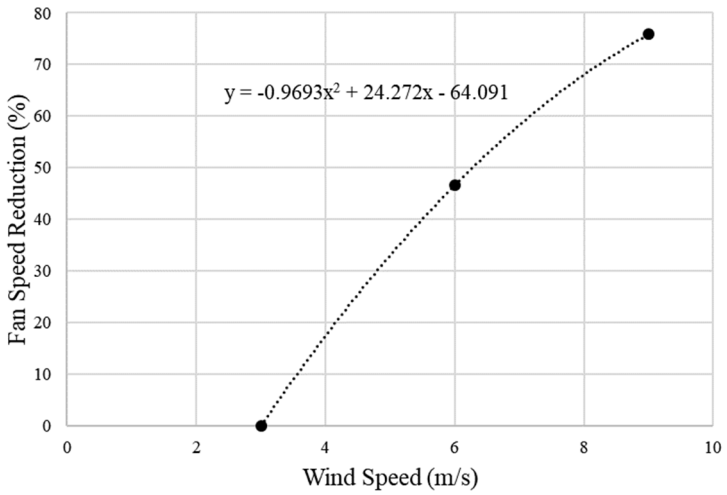


Figure 34: Fan speed reduction with varying wind speeds

The author would like to reiterate that several assumptions were made during this process, including simplifications by using the averaged air volume flow rates across both condenser and mixed cells in the system to develop the mapping, as well as applying the relationship found by Engelbrecht [20] to a different utility-scale ACC system. The method was used as a simplification and last resort due to lack of existing site data including wind speeds for the case study ACC. The author is aware that inaccuracies may arise from the methodology used. Future work will need to

be completed to strengthen the relationship between wind speeds and air volume flow rate reduction for the thermofluid model. However, to develop a condition monitoring platform that can take the wind speed as an input parameter, the above simplifications had to be made. Moreover, the existing mapping between wind speeds and fan speed reduction acts as an initial framework, which can allow for further improvement using CFD data or site measurements.

The input parameter specified to the thermofluid network model regarding wind speeds was a fan speed reduction. The range of wind speeds used in the DOE specified in Figure 32, between $2.5-8.2\ m/s$, was generated through 10% increments from 10% to 70% fan speed reduction. Randomised sampling was used for the wind speeds to sample each increment equally to cover the sample range.

For the wind angle input parameter, wind angles were split into cardinal and ordinal directions; north, north-east, east, south-east, south, south-west, west, and north-west. The wind angle influenced the cells which were affected by wind. With a similar approach to that used in the wind effects study in Section 4.3, only the edge cells in the specified wind direction would have a reduced air volume flow rate (through a reduced fan speed). The actual ACC system was positioned with approximately an 11° offset to compass directions, and for simplification, the ACC system direction was aligned with compass directions.

Figure 35 shows the wind-affected cells for north, south-west, and east wind directions. For cardinal directions (N, E, S, W), the wind-affected cells were a single ACC street (as shown for a north wind or a south wind) or a single column of cells (as shown for an east wind or west wind). For an ordinal wind direction (NE, SE, SW, NW), the affected cells were both a street and column of cells that were incident to the direction of the wind. An example is shown for a south-west wind in Figure 35.

The wind angle input parameter specified which cells were affected by wind. All wind-affected cells would then experience the same fan speed reduction (based on the wind speed input parameter). It is important to note that these assumptions were made as simplifications to incorporate wind effects into the process. In reality, wind effects may be prevalent for inner cells (as opposed to just edge cells) and may not affect all cells equally. Additionally, without a CFD model to properly incorporate the effect of wind speeds into the model nor useable site data for validation of wind effects, the accuracy of the model is significantly limited in this regard. Nevertheless, wind effects were still included in the data-driven surrogate modelling process to act as a surrogate for the thermofluid network model.

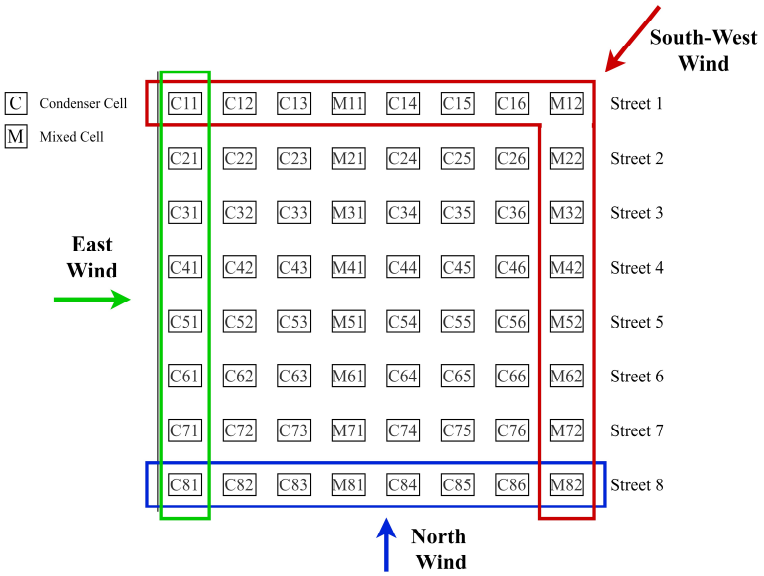


Figure 35: Wind-affected cells with various wind directions

As mentioned previously in this section, the main aim of the data-driven surrogate model was to produce results on a system-wide level rather than on a cell-by-cell basis. For each input sample solved by the thermofluid network model, seven output parameters were recorded, shown in Table 22.

Table 22: Input parameters passed to the thermofluid network model and output parameters recorded

Input Parameters	**Output Parameters**
Ambient Air Temperature (°*C*)	Backpressure (*kPa*)
ACC Inlet Steam Quality	Total Condenser Cell Heat Rejection Rate (*MW*)
ACC Inlet Steam Flow rate (*kg* / *s*)	Total Mixed Cell Heat Rejection Rate (*MW*)
Number of Streets Switched Off (0, 1, 2)	Total Condenser Cell Air Mass Flow Rate (*kg* / *s*)
Wind Angle (Cardinal/Ordinal directions)	Total Mixed Cell Air Mass Flow Rate (*kg* / *s*)
Wind Speed (*m* / *s*)	Total Condenser Cell Fan Motor Power (*kW*)
	Total Mixed Cell Fan Motor Power (*kW*)

The data generation process was automated through a Python API link with Flownex® SE 2019. Automation allowed for data generation to be split across multiple computers to reduce the wall time of the simulations. Excel spreadsheets were used to pass each input sample to Flownex® SE 2019. Flownex® SE would then solve the thermofluid model with the provided inputs as specified in Section 3.7. Once solved, results were then saved, and the next input sample would be passed to Flownex® SE. If a provided input sample resulted in an error in Flownex® SE, the unsolvable input would be flagged, and the next input sample would then be processed. Python code for the API link can be found in Appendix B, and a detailed discussion on the results of the data generation process can be found in Section 5.2.1.

5.1.2 Machine Learning Modelling

In this section, theory and background related to machine learning modelling used will be presented, followed by the methodology in developing the data-driven surrogate model.

Supervised learning is a type of machine learning where training data, including solutions or labels, is fed to a machine learning model. The model uses the labels on the data to provide a mapping error between the inputs in the training data and their associated labels. Supervised learning can be used to tackle classification or regression tasks. For a classification task, the model is initially trained with data that has discrete classes and learns how to classify inputs to a respective class. For a regression task, the machine learning model is trained using inputs with an associated numeric output, to provide a numerical prediction based on provided inputs. The current research involves both a binary classification (only two output labels) and regression tasks. The classification model was used to determine if provided inputs are solvable, and the regression model was used to predict the numerical output values (Table 22).

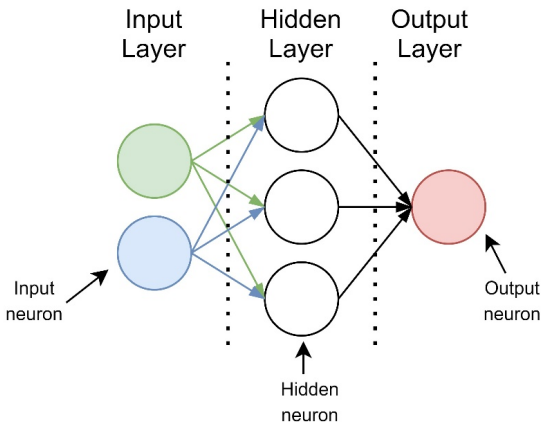


Figure 36: Basic MLP with a single hidden layer

Artificial neural networks (ANNs) are a class of non-linear models typically used for supervised learning tasks. Neurons are the building blocks of ANNs, inspired by the intricately interconnected neural network in the animal brain. Multi-layer perceptron (MLP) networks are a common form of ANNs. In MLPs, neurons are interconnected through several layers, each in turn comprised of multiple neurons. MLPs typically consist of three different types of layers; an input layer, hidden layers, and an output layer [57]. Figure 36 shows a basic MLP with a single hidden layer with a 2/3/1 configuration (input neurons/hidden neurons/output neurons). Deep neural networks (DNNs) typically refer to MLP networks with multiple hidden layers as well as a large number of neurons per layer. DNNs are typically used to model a mapping between highly nonlinear relationships between inputs and the associated outputs.

The number of neurons in the input layer depends on the number of input variables specified to the network. Likewise, for a regression task, the number of predicted output variables determines the number of output neurons. For a binary classification model, a single output neuron is used as the probabilistic output is assigned to one of two classes.

Forward propagation is an algorithm used in MLPs, to pass an input parameter from neurons in the input layer to subsequent neurons in hidden layers and lastly to neurons in the output layer. Weights, w, and biases, b, are two sets of trainable parameters found in MLP networks. These parameters are adjusted through the training process, allowing the network to develop and learn the mapping between provided inputs and outputs. Each neuron is interconnected to all neurons in the previous layer and neurons in the subsequent layer. These connections have weights which are then applied to the signal through the respective neuron, resulting in weighted signals between layers. Biases are added to each neuron separately, as shown below.

MLP networks maximize the conditional probability $P(\hat{y} = Y \mid x = X; \hat{\theta})$, where $\hat{y}$ is the output produced from the MLP network, Y is the target dataset, X is the input dataset, and $\hat{\theta}$ the set of trained parameters, including weights and biases [57]. Before optimising and training network parameters, $\hat{\theta}$ is randomly initialised.

Let a_j^l denote the output of the j-th neuron on the l-th layer of a standard feed-forward MLP. The $l-1$-th layer has d_{l-1} neurons. An updating equation can be used to then define the network structure [58]:

$$a_j^l = \sigma_l\left(\sum_{k=1}^{d_{l-1}} a_k^{l-1} w_{kj}^l + b_j^l\right) \tag{27}$$

where w_{kj}^l is the kj-th weight parameter linking the k-th neuron in the previous layer, $l-1$, with the j-th neuron in layer l.

The summed-product of the k-th neuron output in the previous layer and the linking weight, w_{kj}^{l}, is added to the bias term for the j-th neuron, b_j^l. This is then summed for each neuron in the previous layer. The result is then passed through a chosen activation function for the layer l, which is explained further on.

Figure 37 shows the forward propagation updating equation for the j-th neuron in a basic MLP network with two neurons in each hidden layer, $d_{l-1} = d_l = 2$, and three hidden layers. For parameterization, the third layer was referred to as layer l as described in the above paragraph. For visual clarity, only the bias term for the j-th neuron is shown; however, each neuron would have its own bias parameter. For the j-th neuron, the weights from the two neurons in the previous layer, w_1^l for the red arrow and w_2^l for the blue arrow, would be used in Eq. (27) above. Likewise, this applies to the outputs of the neurons in the second layer, a_1^{l-1} in red and a_2^{l-1} in blue. The same process would apply for bigger networks with more hidden layers and neurons per layer. In the proposed work, the same number of neurons would be used for each hidden layer.

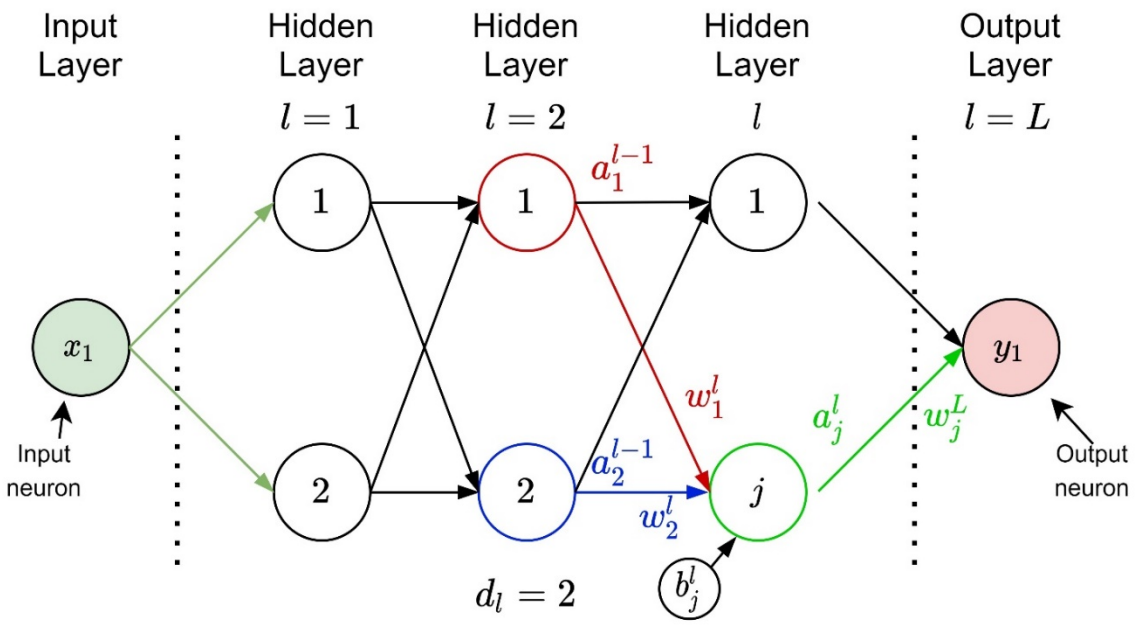


Figure 37: Forward propagation for j-th neuron of a basic MLP network

Activation functions, $\sigma_l(\,)$, are used to introduce non-linearity to MLPs. The output signals from specific neurons are 'activated' and passed through the activation function specified for a given layer. Several activation functions are typically used for MLP networks, such as linear, sigmoid, hyperbolic-tangent, and ReLU functions to name a few. The choice of activation for the output layer

is dependent on the task, regression or classification, while activation functions for input and hidden layers generally are not restricted as such. In this work, a ReLU activation function was used for all hidden layers throughout the machine learning modelling process. Let z_j^l be a linear component, derived from the terms passed to the activation function $\sigma_l(\,)$ in Eq. (27):

$$z_j^l = \sum_{k=1}^{d_{l-1}} a_k^{l-1}\, w_{kj}^l + b_j^l \tag{28}$$

The ReLU activation can then be defined as [57]:

$$a_j^l = \sigma_{ReLU}(z_j^l) = \begin{cases} z_j^l & \text{if} \quad z_j^l > 0 \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

The ReLU function passes the non-zero signal if $z_j^l > 0$. Regarding activation functions for the output layer, the activation function choice depended on the specific task at hand (regression or classification). A linear activation was used for the output layer for the regression model:

$$a_j^L = \sigma_{Linear}(z_j^L) = z_j^L \tag{30}$$

For a regression MLP network, a numerical prediction is produced at the output layer; thus, a linear activation function is typically used [57]. With regards to a binary classifier, a sigmoid activation on the output layer scaled the output to lie between 0 and 1, in turn, assigned the output to a respective class out of two possible classes [57].

$$a_j^L = \sigma_{sigmoid}(z_j^L) = \frac{1}{1+e^{-z_j^L}} \tag{31}$$

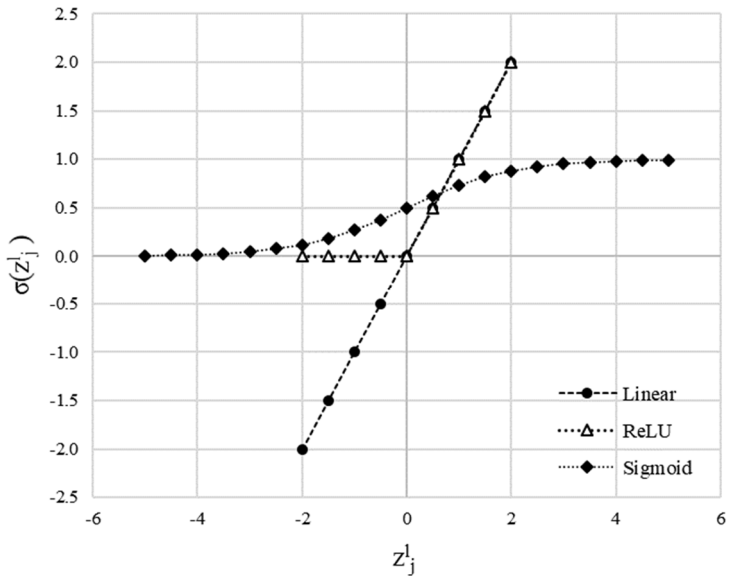


Figure 38: Linear, ReLU, and Sigmoid activation functions

Figure 38 shows the linear, ReLU, and sigmoid activation functions with the respective effect on the scaled outputs. To successfully optimise the network parameters, $\hat{\theta}$, a measure of prediction performance was introduced through the form of a cost function. For regression tasks, the output of a network may take on any value within an infinite range of possible outcomes. Consequently, a measure for the quality of a prediction can be derived as the distance between the prediction and observed output from the training dataset. This is known as the mean-squared-error (MSE) cost function [57]:

$$C_{MSE}(\hat{y},Y)=\frac{1}{n}\sum_{i=1}^{N}(\hat{y}_i-Y_i)^2 \tag{32}$$

Eq (32) calculates the MSE using the predictions from the MLP network, $\hat{y}$, and the observed outputs from the training dataset, Y. With a linear activation function on the output layer for regression tasks, for each output variable, $\hat{y}_i=a^L$. For each sample in the dataset, the difference between the predicted output and the observed output is squared. The average is then found over n samples in the observed training dataset. For multiple output variables, the average MSE is found for each variable and averaged over the number of output variables.

For classification tasks, the output of the network takes on discrete values from a known finite set of classes. Thus, a cost function can be formulated in terms of probabilities of taking on a particular class. The use of a sigmoid activation function on the output layer, which scaled the output to lie within 0 and 1, allowed for the output to be interpreted as probabilities. Therefore, a binary cross-entropy error cost function, where the true classes $Y\in\{0,1\}$, can be defined as [57]:

$$C_{CE}=-\frac{1}{n}\sum_{i=1}^{N}(Y_i\log(\hat{y}_i)+(1-Y_i)\log(1-\hat{y}_i)) \tag{33}$$

By using a sigmoid activation function on the output layer, the prediction from the network, $\hat{y}_i$, is essentially a probability estimate, $\hat{y}_i=P(\hat{y}_{i_{class}}=1)$, estimating if a prediction belongs to one class.

The backpropagation algorithm developed by Rumelhart et al. [59] was used to optimize network parameters. After a single forward propagation was completed, the cost function was used to determine how well predictions match the target data. Backpropagation refers to the calculation of the gradient of the cost function with respect to each network parameter (error gradients), propagating backwards from the cost function at the output layer to preceding layers till the input layer is reached.

Using the previously defined linear component from Eq. (28), and a working gradient of the cost function with respect to the linear component:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{34}$$

General expressions can be derived for the required gradients with respect to various network parameters. A detailed breakdown of the equations presented below can be found in [57], [58].

The working gradient can be defined, starting from the output layer [58]:

$$\begin{aligned} \delta_j^L &= \sum_{k=1}^{d_L=q} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L) \end{aligned} \tag{35}$$

For preceding layers [58]:

$$\begin{aligned} \delta_j^{l-1} &= \frac{\partial C}{\partial z_j^{l-1}} \\ &= \sum_{k=1}^{d_l} \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial z_j^{l-1}} \\ &= \sum_{k=1}^{d_l} w_{jk}^l \delta_k^l \sigma'(z_j^{l-1}) \end{aligned} \tag{36}$$

Importantly, Eq. (36) defines the working gradient in a preceding layer using the weights and working gradient in the subsequent layer, as well as the known linear component from forward propagation in the preceding layer. Thus, all parameters in Eq. (36) are known after forward propagation. For the working gradient with respect to biases and weights [58]:

$$\begin{aligned} \frac{\partial C}{\partial b_j^l} &= \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial b_j^l} \\ &= \delta_j^l \end{aligned} \tag{37}$$

$$\begin{aligned} \frac{\partial C}{\partial w_{kj}^l} &= \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{kj}^l} \\ &= a_k^{l-1} \delta_j^l \end{aligned} \tag{38}$$

Notably, Eq. (35) requires the partial derivative of the cost function with respect to the output of the final layer. For regression models with an MSE cost function, replacing $\hat{y}$ with a_j^L in Eq. (32):

$$\frac{\partial C_{MSE}}{\partial a_j^L} = \frac{2}{n} \sum_{i=1}^{N} (a_j^L - Y_i) \tag{39}$$

Likewise, for a classification model with a cross-entropy loss function:

$$\begin{aligned}\frac{\partial C_{CE}}{\partial a_j^L} &= \frac{2}{n}\sum_{i=1}^{N}\frac{Y_i}{a_j^L} - \frac{1-Y_i}{1-a_j^L} \\ &= \frac{2}{n}\sum_{i=1}^{N}\frac{Y_i - a_j^L}{a_j^L(1-a_j^L)}\end{aligned} \tag{40}$$

Similarly, the derivatives of activation functions are also required. The derivative for a linear activation function is a constant, however, for a ReLU activation function:

$$\sigma'_{ReLU}(z_j^l) = \begin{cases} 1 & \text{if} \quad z_j^l > 0 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

For a sigmoid activation function for the output layer in a classification model:

$$\sigma'_{sigmoid}(z_j^L) = \frac{e^{-z_j^L}}{(1+e^{-z_j^L})^2} \tag{42}$$

The equations described above, therefore, allowed for error gradients to be computed for each network parameter. Network parameters were then adjusted using the known error gradients. Many methods such as Adam, RMSProp, Adagrad, and SGD have been used to adjust network parameters. The current work uses the Adam algorithm for adaptive learning rate optimisation [60]:

$$\begin{aligned} &g_t \leftarrow \nabla_{\bar{\hat{\theta}}} C_t\left(\bar{\hat{\theta}}_{t-1}\right) \\ &\bar{m}_t \leftarrow \beta_1 \bar{m}_{t-1} + (1-\beta_1) g_t \\ &\bar{v}_t \leftarrow \beta_2 \bar{v}_{t-1} + (1-\beta_2) {g_t}^2 \\ &\hat{\bar{m}}_t \leftarrow \frac{\bar{m}_t}{1-{\beta_1}^t} \\ &\hat{\bar{v}}_t \leftarrow \frac{\bar{v}_t}{1-{\beta_2}^t} \\ &\bar{\hat{\theta}}_t \leftarrow \bar{\hat{\theta}}_{t-1} - \eta \hat{\bar{m}}_t \otimes \left(\sqrt{\hat{\bar{v}}_t} + \varepsilon\right)^{-1} \end{aligned} \tag{43}$$

For a given iteration t, where a forward propagation step has taken place at the previous iteration, the algorithm first computes all error gradients for the defined cost function with respect to a vector of known parameters in the network, $\bar{\hat{\theta}}_{t-1}$. Initially at timestep $t = 0$, a first moment vector, $\bar{m}_0$, and second moment vector, $\bar{v}_0$, are equal to 0. The first moment vector, $\bar{m}_t$, is calculated using the first decay hyperparameter, β_1, fixed at $\beta_1 = 0.9$, as well as the calculated error gradients g_t. Likewise, the second moment vector, $\bar{v}_t$, is calculated using the second decay hyperparameter, β_2, fixed at $\beta_2 = 0.999$. The first and second moment vectors are then bias-corrected using the decay

hyperparameters to the power t. Lastly, an updated vector of parameters, $\overline{\hat{\theta}}_t$, is calculated using the old vector of hyperparameters, $\overline{\hat{\theta}}_{t-1}$, a learning rate hyperparameter η, the bias-corrected moment vectors, and $\varepsilon = 10^{-8}$ for numerical stability. The learning rate, η, is adjusted through the training process and is discussed in Section 5.1.3.

Through the combination of forward propagation, the cost function calculation, backpropagation and parameter optimisation using Adam, the network parameters can be trained to minimise the loss calculated from the cost function. This process occurs during a training and validation phase, where the network parameters are trained using a training dataset to provide the best prediction. Once a network has been trained, validation takes place where several hyperparameters are tuned based on unbiased performance on a validation dataset. Once hyperparameters are selected for the final MLP network, a testing phase is where the trained and tuned MLP network is used to provide predictions based on unseen inputs which were not provided to the MLP network during the training and validation phase. The testing phase allows for unbiased performance analysis of the developed MLP network.

Mini-batches are often used to pass training samples through to the MLP network. Mini-batches consist of a fixed number of data samples forming subsets from the primary dataset. For each mini-batch passed to the MLP network, the process described above is completed (forward propagation, cost function calculation, backpropagation, and parameter update). The next mini-batch is then passed to the MLP network, till the entire training dataset has been passed through. Each complete pass of mini-batches totalling the entire dataset is known as an epoch.

Mini-batches can benefit from the use of GPU computing, and allow for more updates to network parameters within a single epoch, as compared to passing the entire dataset in a single batch, known as batch gradient descent [57]. Furthermore, mini-batch gradient descent provides a measure of regularization during the training of the network parameters. Regularization is where a model is constrained to make it simpler. There are several regularisation methods, such as early stopping as well as weight decay, where the introduction of a penalty term in the cost function limits weight magnitudes [57], [61].

The size of the mini-batch is a tuneable hyperparameter, known as the batch size. Additionally, the number of epochs that the model is trained for is also a tuneable hyperparameter. Early stopping is a technique to limit the epoch size to avoid overfitting the training dataset and was used in the present work for regularisation [57]. Dropout and batch normalization [57] are also two additional techniques used for regularization and were tested during the modelling process but not used in the final developed models.

Feature scaling is commonly used to pre-process datasets used in machine learning models. Feature scaling ensures that all input parameters are scaled to the same range, as ML algorithms generally do not perform well with inputs that have vastly varied scales [57]. Min-max scaling was used in this work to ensure that all input parameters were scaled between 0 and 1. Each input parameter would be scaled separately using [57]:

$$\overline{X}_{std} = \frac{\overline{X} - X_{min}}{X_{max} - X_{min}} \tag{44}$$

Each input parameter was scaled, where $\overline{X}$ is a vector of input samples for one input parameter, $X_{min/max}$ is the minimum and maximum input sample, and $\overline{X}_{std}$ the scaled vector of input samples for one input parameter. Similarly, the labelled output samples are also feature scaled using a min-max scaler. To extract unscaled predictions from the model, Eq. (44) is rearranged to make $\overline{X}$, or $\overline{Y}$ in the case of outputs, the subject of the equation, with other parameters saved from the initial scaling process. With the use of a ReLU activation function, min-max scaling also safeguarded against exploding error gradients throughout backpropagation.

Cross-validation is a technique used during the training and validation phase. K-fold cross-validation involves splitting the training dataset into K subsets called 'folds'. Six-fold cross-validation refers to the training set split into six-folds, which was used in the present work. For a given MLP network with six-fold cross-validation, the model was trained using five-folds and validated using one-fold. This was performed six times, where each fold was used once as the validation fold. Consequently, an average validation loss was found over the six training instances, providing a more accurate and less biased measure of performance compared to using a single validation set. Cross-validation was used in tandem with a hyperparameter search. A hyperparameter search referred to finding optimum hyperparameters for a given model to produce the best predictions. It also determined how to structure the MLP network in terms of the number of hidden layers as well as neurons per hidden layer. The use of cross-validation with a hyperparameter search allowed for the best model to be selected without compromising the unbiased test set, determining the performance of an MLP network based on unseen data.

Figure 39 shows an example of six-fold cross-validation with a hyperparameter search across three different models. Each model uses different hyperparameter values (such as different amounts of hidden layers, learning rates, and more). For each model, cross-validation was performed during the training/validation phase. The average validation loss (evaluated using the cost function) across the six folds was found and used to select the best performing model. Lastly, the selected model was then used to predict outputs based on the test set and compared to the actual outputs in the test set.

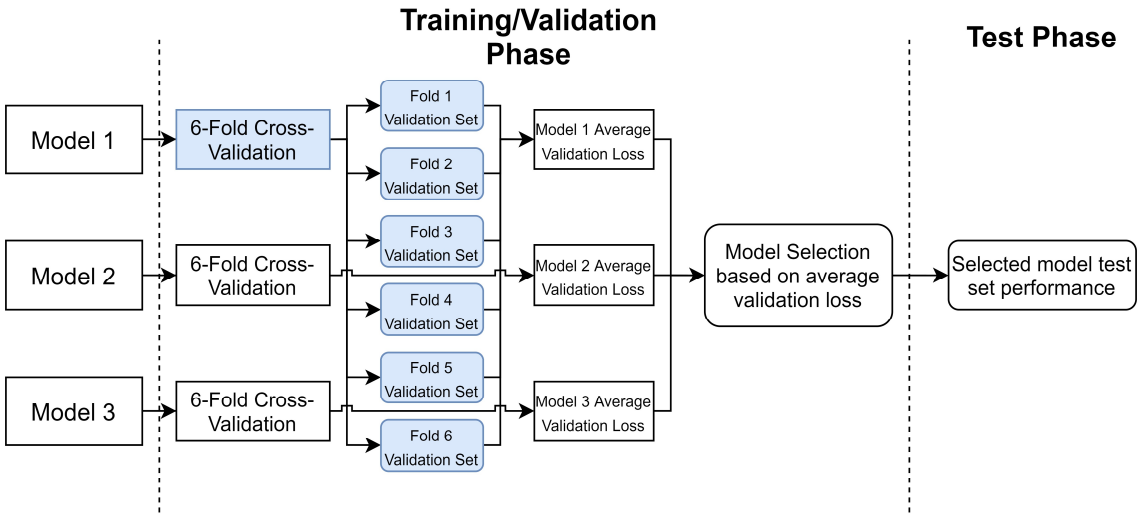


Figure 39: Training/validation and test phase for a hyperparameter search with three models

5.1.3 Integrated Model Architecture Development

A single final combined dataset was used during the machine learning modelling process. The benefit of using a single combined dataset was that only one MLP network was required. However, using the initial three developed datasets (Figure 32) with their own respective MLP networks was also explored. It was found that there was no performance improvement over a single combined dataset with one MLP network compared to three individual MLP networks with their respective datasets.

This combined dataset contained inputs and results from the thermofluid network model for the three different datasets as specified in Section 5.1.1 and Figure 32. Therefore, a single set of input variables were present for each data sample used in the machine learning modelling process. To avoid extrapolation between datasets due to combining input variables not present for a given dataset, standard or fixed inputs were specified. Regarding the input parameters missing between various datasets in Table 19, Table 23 shows fixed inputs for the three datasets.

A hyphen represented parameters without fixed values. As an example, dataset 1 did not account for switching off streets, nor did it account for wind effects. Consequently, in the final combined dataset, 0 was specified for the "number of streets switched off" parameter for all inputs from dataset 1. Likewise, for the wind angle and wind speed, a default 'N' and $2.5\,m/s$ was specified in the final combined dataset for all input samples from dataset 1. This then followed on in the final combined dataset for various input parameters in the other two datasets.

Table 23: Fixed inputs for the three developed datasets

Dataset 1	Value	Dataset 2	Value	Dataset 3	Value
Ambient Air Temperature (°*C*)	-	Ambient Air Temperature (°*C*)	-	Ambient Air Temperature (°*C*)	-
ACC Inlet Steam Quality	-	ACC Inlet Steam Quality	-	ACC Inlet Steam Quality	-
ACC Inlet Steam Flow rate (*kg* / *s*)	-	ACC Inlet Steam Flow rate (*kg* / *s*)	-	ACC Inlet Steam Flow rate (*kg* / *s*)	-
Number of Streets Switched Off	0	Number of Streets Switched Off (1,2)	-	Number of Streets Switched Off	0
Wind Angle	N	Wind Angle	N	Wind Angle (Cardinal/Ordinal directions)	-
Wind Speed (*m* / *s*)	2.5	Wind Speed (*m* / *s*)	2.5	Wind Speed (*m* / *s*)	-

Table 24: Binary classifier and regression MLP networks input and output parameters

Binary Classifier MLP Network		**Regression MLP Network**	
Input Parameters	Output Parameters	Input Parameters	Output Parameters
Ambient Air Temperature (°*C*)	Solvable by thermofluid network model (Y/N)	Ambient Air Temperature (°*C*)	Backpressure (*kPa*)
ACC Inlet Steam Quality		ACC Inlet Steam Quality	Total Condenser Cell Heat Rejection Rate (*MW*)
ACC Inlet Steam Flow rate (*kg* / *s*)		ACC Inlet Steam Flow rate (*kg* / *s*)	Total Mixed Cell Heat Rejection Rate (*MW*)
Number of Streets Switched Off		Number of Streets Switched Off	Total Condenser Cell Air Mass Flow Rate (*kg* / *s*)
Wind Angle		Wind Angle	Total Mixed Cell Air Mass Flow Rate (*kg* / *s*)
Wind Speed (*m* / *s*)		Wind Speed (*m* / *s*)	Total Condenser Cell Fan Motor Power (*kW*)
			Total Mixed Cell Fan Motor Power (*kW*)

In the present work, both a regression task and classification task were addressed during the data-driven surrogate modelling process using MLP networks, as shown in Table 24. The regression model was the main focus, producing numerical performance predictions based on ambient and ACC operating inputs. Classification, in the form of a binary classifier, was used to predict whether a provided set of inputs were solvable by the thermofluid network model. This was concerning the scope of the three datasets generated earlier (Figure 32). Classification was an important

consideration to avoid extrapolation from the dataset used to train the regression model, avoiding inaccurate predictions based on inputs that lay outside the scope of the regression model. The output labels for the classifier consisted of whether the respective input parameters were solvable (1) or not solvable (0) by the thermofluid network model.

The regression MLP network used the final combined dataset, as mentioned earlier. The binary classifier MLP network used the inputs from the final combined dataset, with the output parameter marked as being solvable. The classifier dataset also contained augmented inputs. These augmented inputs consisted of input parameters that were not solvable by the thermofluid network model, as well as additional artificially generated inputs that were outside the scope of the three developed datasets (Figure 32), marked as not solvable. Details on the classifier dataset can be found in Section 5.2.1.

Figure 40 shows the detailed breakdown of the DDS model design process, split into data generation, the training and validation phase, and the testing phase. With regards to the hyperparameter search space, typically a grid search is used. A grid search involves specifying a range of discrete values for each hyperparameter, shown in Table 25, forming an MLP network for each combination of hyperparameters. In the present work, a grid search was initially used as a sensitivity analysis between hyperparameters and validation losses. However, a manual search was used as the primary hyperparameter tuning method, where hyperparameter values were tuned iteratively, which led to the final model selection for both the regression network MLP and the binary classifier network MLP.

Table 25: Hyperparameters tuned in model development

Hyperparameters	
Batch size	Hidden layers
Learning rate	Neurons per hidden layer
Epochs	

The datasets used for the regression and binary classifier MLP networks respectively were split into 80% for training and 20% for testing. The training sets were then used during the six-fold cross-validation process to train the networks. For the regression MLP network, 11 different networks were configured through the hyperparameter tuning process and used in the validation phase. For the binary classifier MLP network, eight different networks were configured. A final model for both the regression and binary classifier network was then selected. These models were then evaluated using the test set.

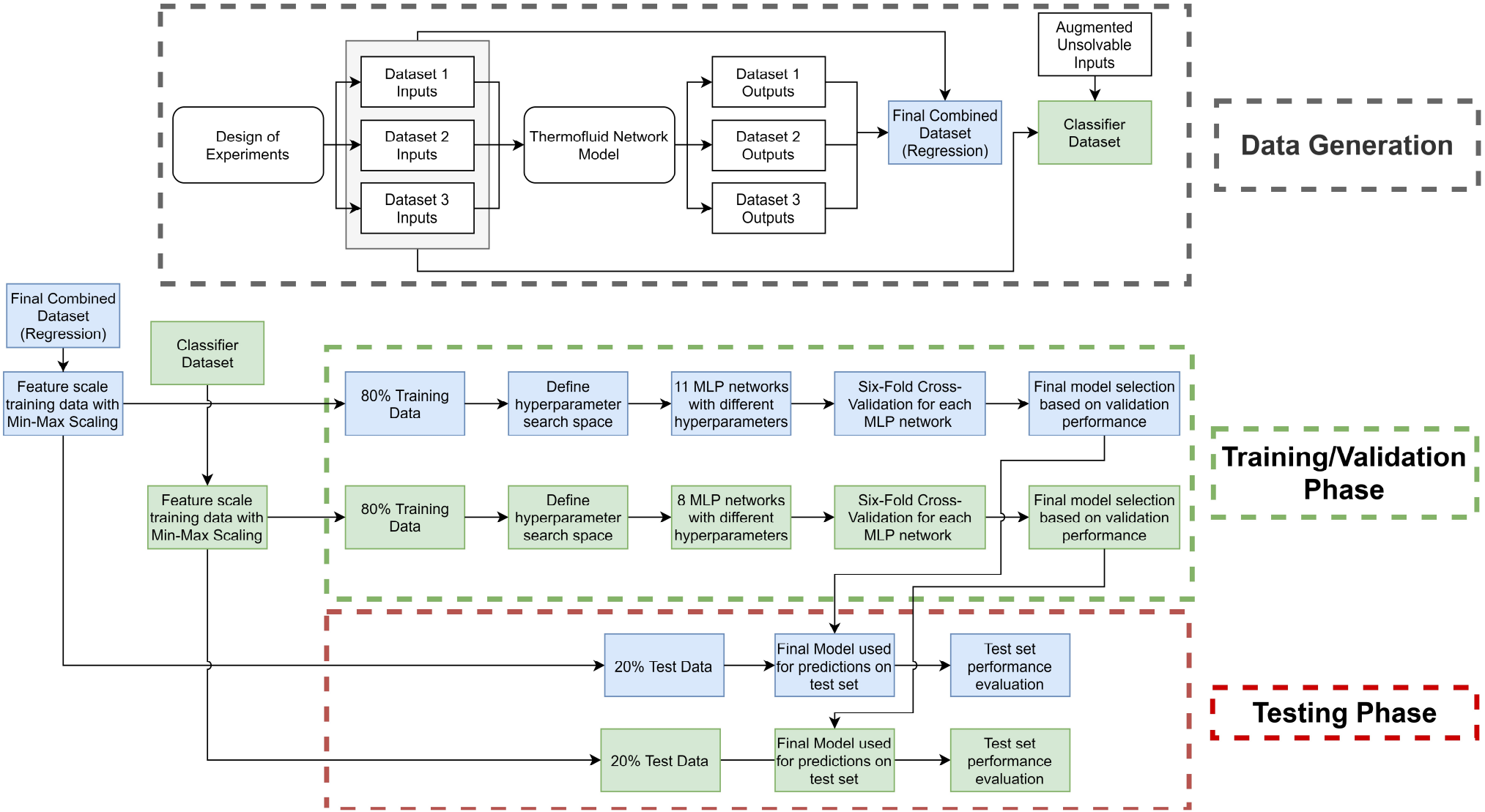


Figure 40: Integrated DDS model design process

Table 26 and Table 27 show the different MLP networks developed through the hyperparameter tuning process for the regression and binary classifier networks, respectively. A coarse grid search was initially used to investigate the effects of various hyperparameters on validation performance. Subsequently, a manual search was then used to tune hyperparameters to arrive at the best performing network iteratively. Detailed results and discussions regarding the hyperparameter search as well as test set performances can be found in Section 5.2.2 and 5.2.3.

Table 26: 11 Regression MLP networks with respective hyperparameters

MLP Network (R)	1	2	3	4	5	6	7	8	9	10	11
Batch size	128	256	256	64	64	64	64	64	32	64	64
Learning rate	1E-4	1E-4	1E-4	1E-3	1E-3	1E-4	1E-4	1E-4	1E-4	1E-4	1E-4
Epochs	499	990	918	695	803	953	954	500	438	425	500
Neurons per hidden layer	4096	2048	4096	4096	8000	4096	4096	4096	4096	5500	4096
Hidden Layers	2	2	2	2	2	3	4	5	5	5	6

Table 27: Eight binary classifier MLP networks with respective hyperparameters

MLP Network (BC)	1	2	3	4	5	6	7	8
Batch size	64	64	64	64	64	64	64	32
Learning rate	1E-3	1E-3	1E-3	1E-4	1E-3	1E-3	1E-3	1E-3
Epochs	159	75	169	151	81	124	68	65
Neurons per hidden layer	512	1024	1024	1024	512	512	256	512
Hidden Layers	2	2	3	3	3	4	3	3

Figure 41 shows the breakdown of the developed data-driven surrogate model. Inputs were supplied to the model, as shown in Figure 41. These inputs were fed to the binary classifier, which predicted whether the provided inputs were within the scope and solvable by the developed thermofluid network model. If not, the inputs had to be adjusted accordingly to fall into scope.

Solvable inputs were then passed to the regression network, which in turn, provided the ACC performance predictions.

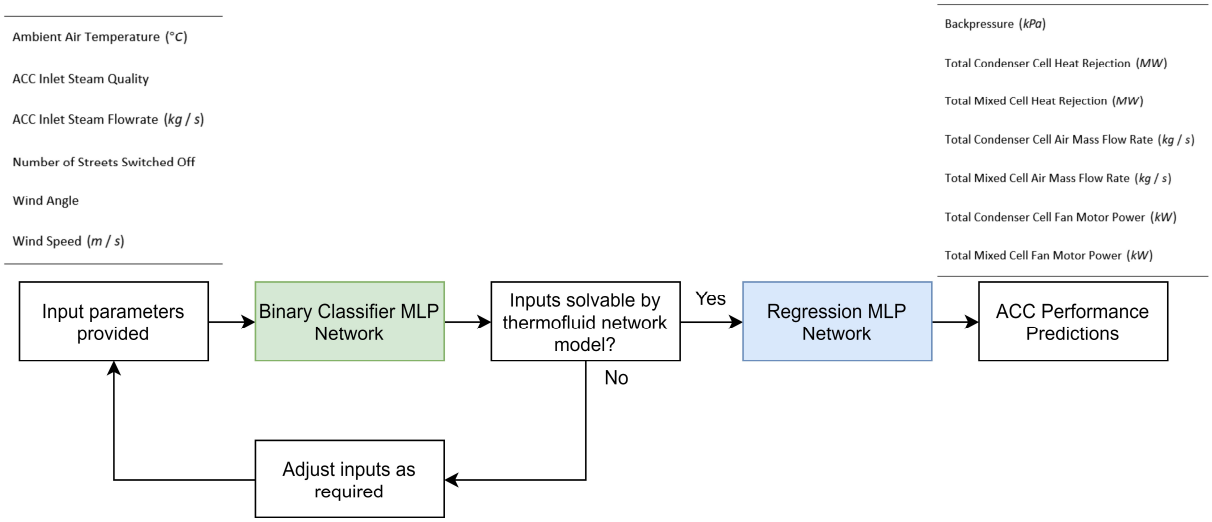


Figure 41: Data-driven surrogate model breakdown

All machine learning modelling was conducted in Python 3.7, using PyTorch 1.6, Skorch 0.8.0, and Scikit-Learn 0.23. PyTorch was used for the fundamental machine learning models. Skorch allowed for Scikit-Learn compatibility with PyTorch models, used mainly for cross-validation and various performance metrics. Code for ML modelling can be observed in Appendix C.

A web-application prototype was developed which hosted the data-driven surrogate model using a Flask API with several HTML templates. The web-app had three main sections; a page for processing a single input sample, a page for processing multiple input samples through a spreadsheet, and lastly a forecasting tool for ACC performance predictions based on forecasted weather data at the ACC system location. The forecasting tool was the focus of the web-app and used the OpenWeatherMap API [62], which provided weather forecasts for five days in three-hour intervals. Details on the web-app development can be found in Section 5.2.5 and Appendix D. The ideology behind the web-app development was to demonstrate the practicality of a data-driven surrogate model and its use in a condition monitoring environment.

5.2 Results and Discussion

This section contains results and discussion for the data-driven surrogate modelling process in the project. The section was split into four subsections; data generation, the regression MLP network, the binary classifier MLP network, and lastly the web-based forecasting tool prototype.

5.2.1 Data generation

In this subsection, the results from the various DOEs are presented. These results were the generated inputs that were passed to the thermofluid network model. Subsequently, the result datasets generated by the thermofluid network model are discussed, along with the final regression model combined dataset and binary classifier dataset.

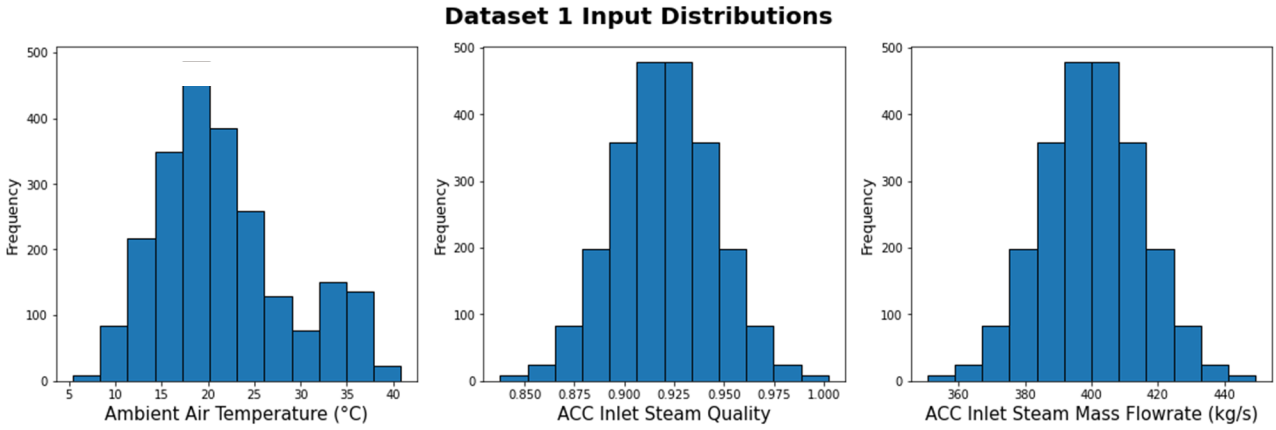


Figure 42: Distribution of generated inputs for dataset 1 (no wind, all fan streets on)

Figure 42 shows an overview of the input samples generated by the DOEs for dataset 1. Normal distributions were observed as expected through the sampling process. The ambient air temperature histogram presented in Figure 42 combined all the seasonal inputs generated. Moreover, the summer DOE for hotter temperatures was included, resulting in a mixture Gaussian distribution with a secondary probability peak observed at approximately $32°C$. A range between $5°C - 40°C$ was generated for the input space of ambient air temperatures. For the ACC inlet steam quality and mass flow rate respectively, a range of $0.85 - 1.00$ and $360 - 440\ kg/s$ was generated.

Figure 43 shows the normalized distribution with the seasonal breakdown for ambient air temperatures in dataset 1. The hotter summer temperatures covered a narrow range compared to the other seasonal DOEs, resulting in a higher normalized frequency. The combination of all the seasonal distributions above resulted in the ambient air temperature plot from Figure 42. An acceptable range of ambient air temperatures across each season was covered in the input space.

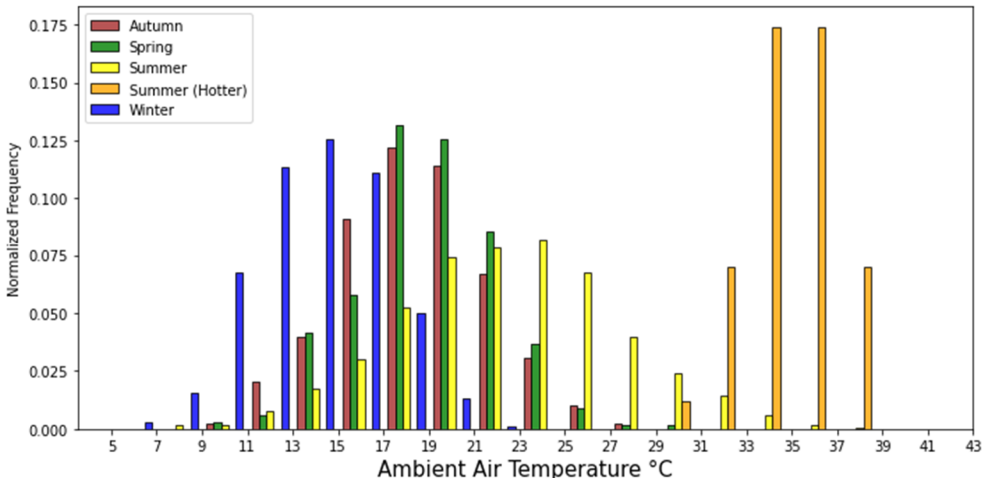


Figure 43: Normalized seasonal ambient air temperature distribution for dataset 1 (no wind, all fan streets on)

Figure 44 shows the generated input space between the three input parameters used for dataset 1. A total of 2300 input data samples were generated covering multiple seasons. There was slight sparsity between the hotter temperatures DOE and the remaining seasonal DOEs at around $30°C$; however, there was still enough sampling in between to cover the input space, as shown later with good generalisation achieved on the test set.

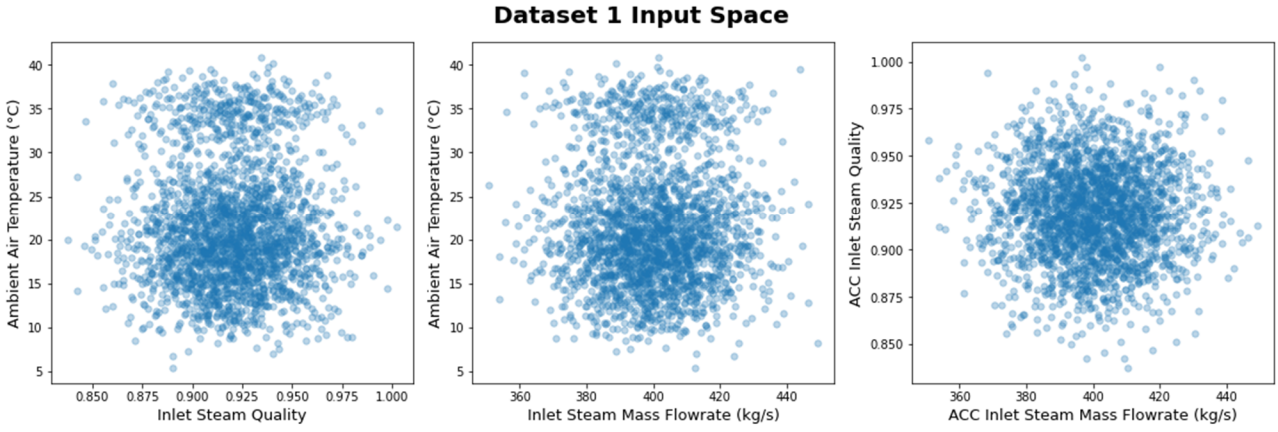


Figure 44: Generated input space for dataset 1 (no wind, all fan streets on)

For dataset 2 a single DOE was used to generate inputs. The same set of inputs were then used for one street switched off as well as two streets switched off in the thermofluid network model. Figure 45 shows the generated inputs distribution for dataset 2. As expected, normal distributions were observed for all three input parameters. The ambient air temperature input distribution was generated using the same mean and standard deviation for the winter DOE from dataset 1.

Consequently, the range of ambient air temperatures was shortened to $5°C - 22.5°C$. The inlet steam quality distribution was also slightly narrower, $0.90 - 0.98$, while the steam mass flow rate remained the same, $360\ kg/s - 440\ kg/s$.

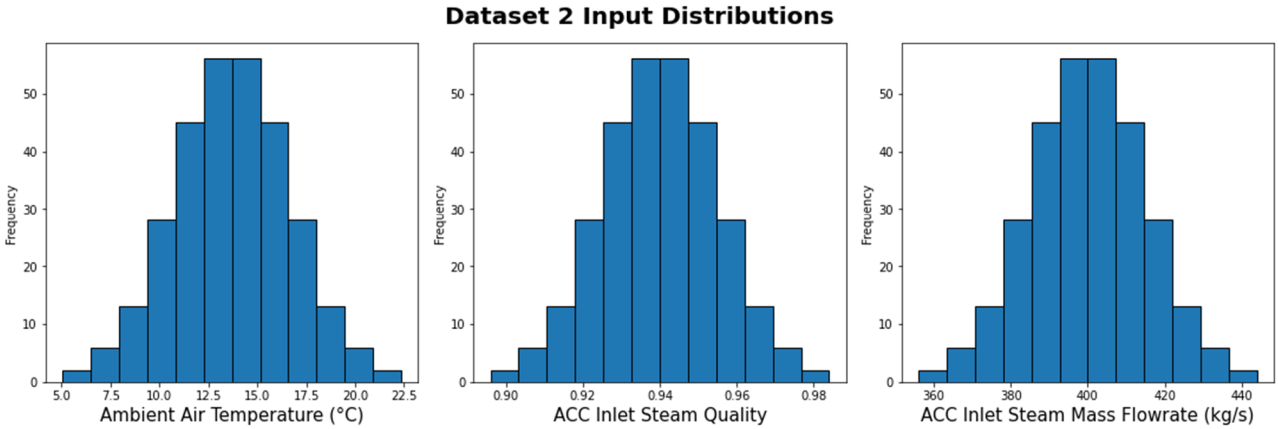


Figure 45: Dataset 2 (fan streets switched off) distribution of generated inputs

Figure 46 shows the generated input space coverage for dataset 2. The 600 combined data points sufficiently covered the input space. Dataset 2 had fewer samples as compared to dataset 1, mainly due to a smaller range of ambient air temperatures; however, enough data samples were generated to cover the input space, as shown later with good generalization when fitting the regression MLP network.

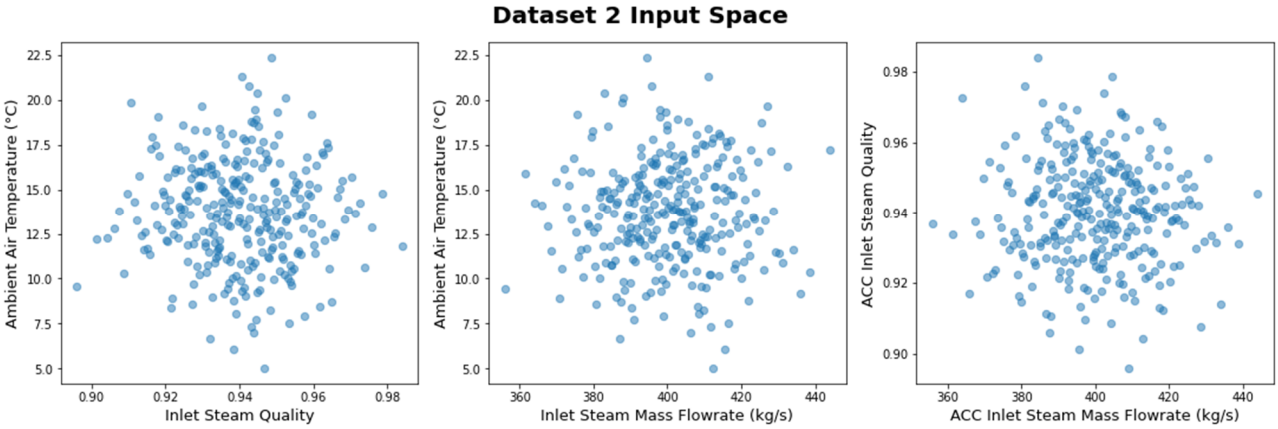


Figure 46: Generated input space for dataset 2 (fan streets switched off)

For dataset 3, a single DOE was used to generate the data considering wind effects. Figure 47 shows the distributions of the generated input parameters, while Figure 48 shows the generated input space between the input parameters. The ambient air temperature input range was developed to

cover the 95% confidence interval of local ambient air temperatures at the ACC system locality, ranging from $7.5°C - 29.5°C$. The ACC inlet steam quality inputs ranged from 0.85-1.00, while ACC inlet steam mass flow rates ranged from $360\,kg/s - 440\,kg/s$. Figure 47 displays the fan speed reduction inputs generated, rather than the wind speed. This was because the fan speed reduction inputs were passed to the thermofluid network model. The wind speeds were specified as inputs to the data-driven surrogate model, which would then be mapped to fan speed reductions before being passed to the regression MLP network. Both the wind angle and fan speed reduction input parameters were randomly sampled. Each discrete value for the wind angle had over 200 samples, with north-east being the most sampled. Likewise, for the fan speed reduction input parameter, all values had over 200 input samples.

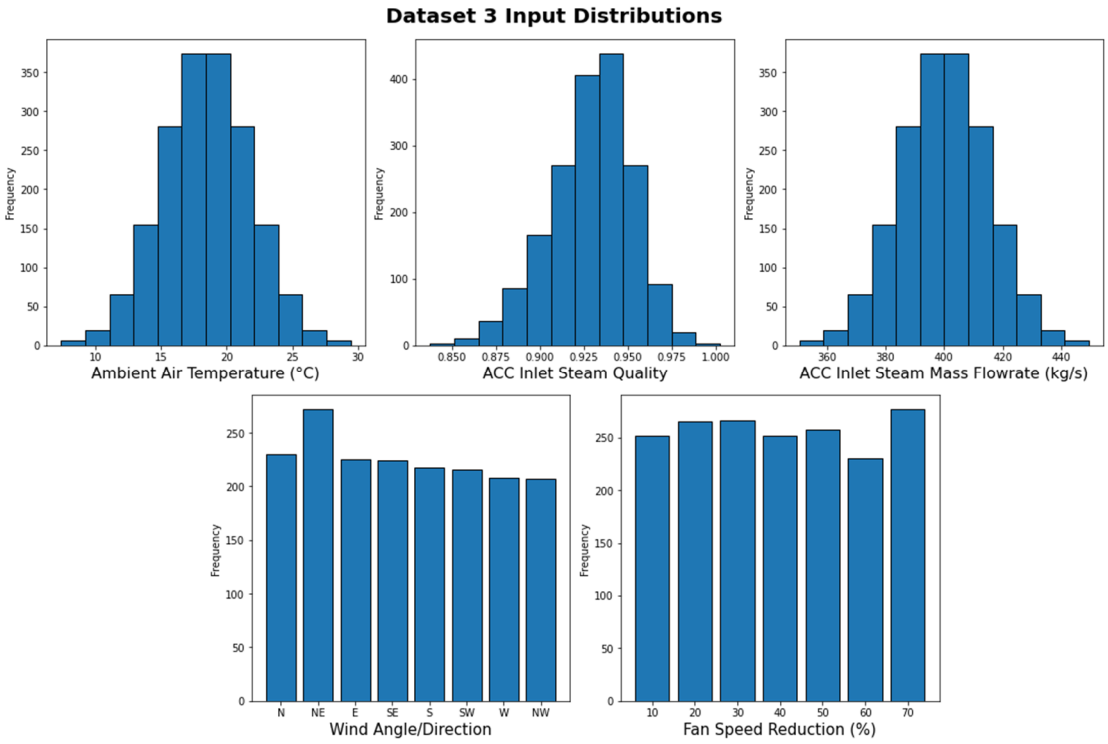


Figure 47: Dataset 3 distribution of generated inputs (wind effects, all fan streets)

From Figure 48, the input space between ambient air temperatures, inlet steam qualities, and inlet steam mass flow rates show sufficient sampling to cover the input ranges mentioned above, with a total of 1800 data samples. When considering the wind angle and fan speed reduction parameters, the graphs in the second row of Figure 48 shows that the sampling covered each discrete value of wind angle and fan speed reduction for the first three input parameters. For each discrete wind angle, the sample space covered each fan speed reduction increment, as seen in the last graph in Figure 48.

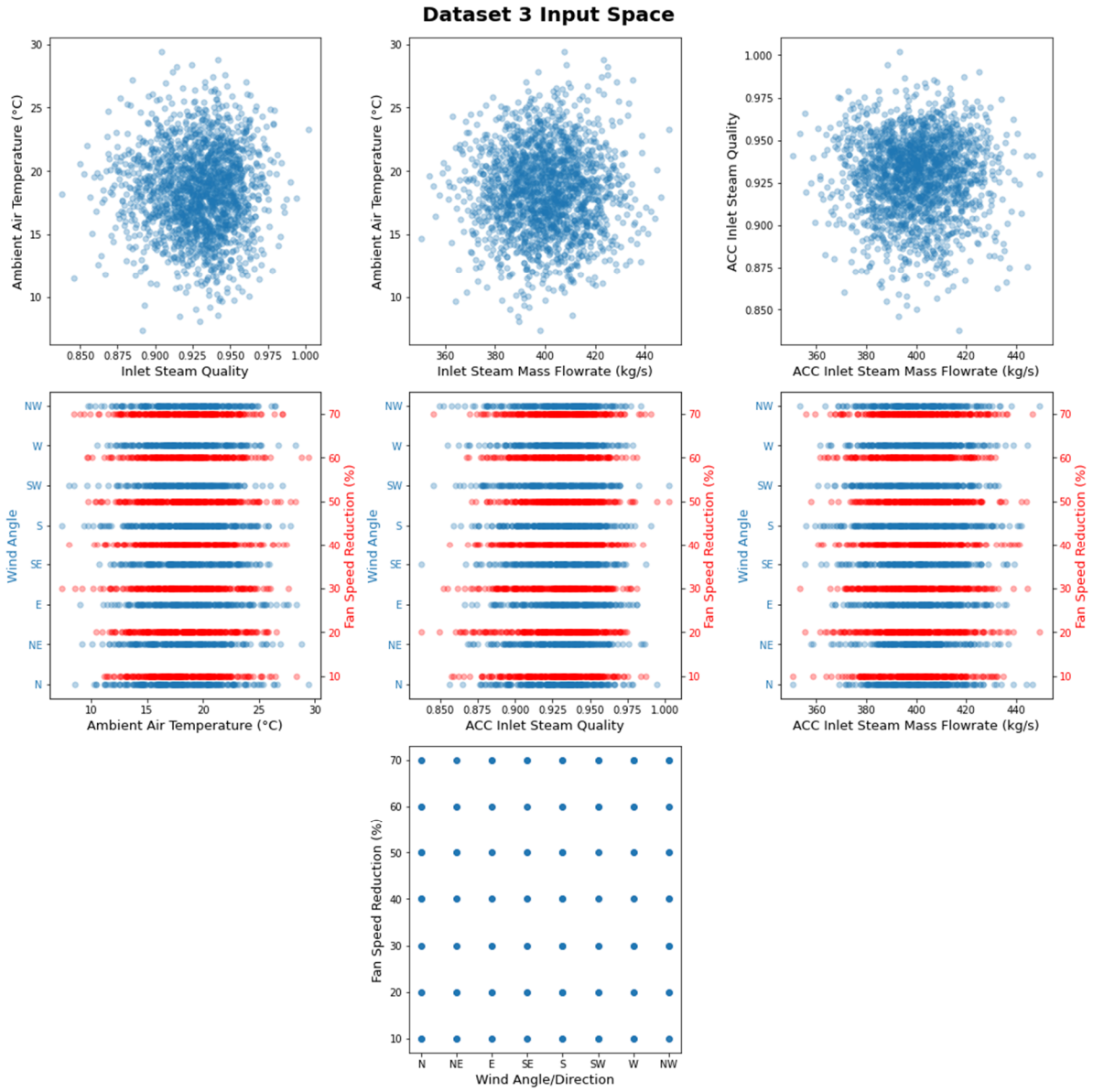


Figure 48: Generated input space for dataset 3 (wind effects, all fan streets)

The figures presented in this subsection thus far have shown that the data generated through LHS sufficiently covered the input space prescribed in Section 5.1.1. A total of 4700 input samples were generated across the three datasets. These input samples were then solved by the thermofluid network model with the generated result parameters (Table 22) saved for each processed input sample. The total time taken to generate the data, including manual pre-processing and post-processing, was approximately three weeks.

A brief discussion of results generated by the thermofluid network using the input samples is presented below. Figure 49 shows the backpressure against ambient air temperature, inlet steam

quality, and inlet steam mass flow rate, respectively for dataset 1 (all streets operating without wind effects). A clear non-linear trend was observed between backpressure and ambient air temperature, with the backpressure increasing non-linearly at higher ambient air temperatures, above $35°C$. On the other hand, there was no visible trend in backpressures with increasing inlet steam qualities nor increasing steam mass flow rates. Therefore, a change in ambient air temperatures was the main driving force behind a backpressure change.

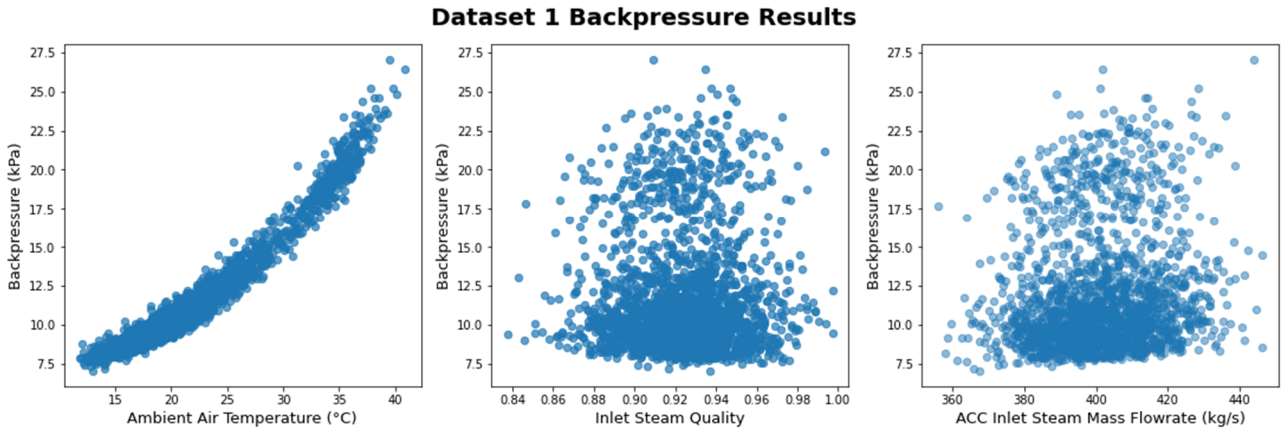


Figure 49: Backpressure with ambient air temperatures, inlet steam qualities, and inlet steam flow rates for dataset 1

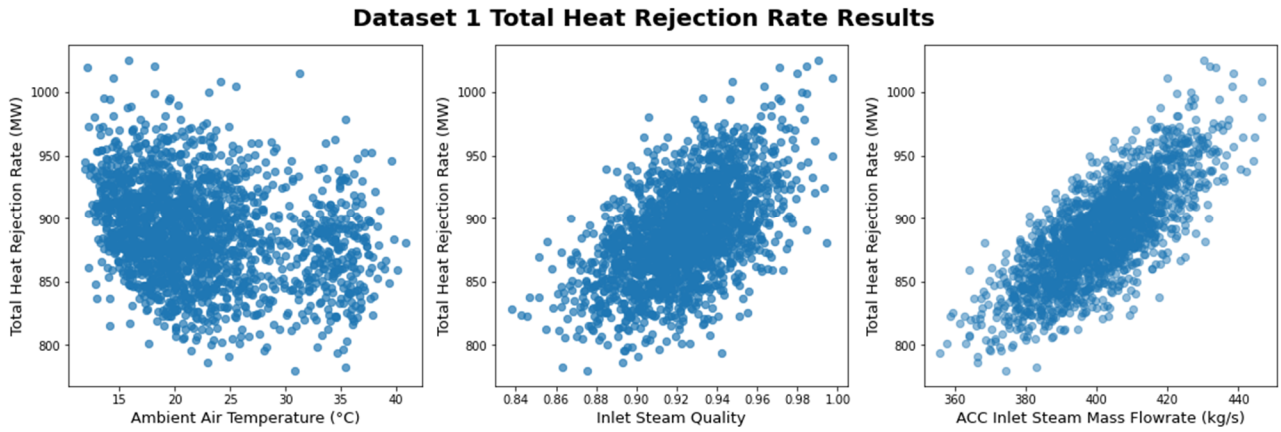


Figure 50: Total heat rejection rates with ambient air temperatures, inlet steam qualities, and inlet steam flow rates for dataset 1

Figure 50 shows the total system heat rejection rate against ambient air temperature, inlet steam quality, and inlet steam mass flow rate for dataset 1. Unlike the trends observed for the backpressure predictions in Figure 49, each of the three input parameters affected the total heat rejection rate. A lower ambient air temperature resulted in slightly higher heat rejection rates, due to the increased temperature difference across the steam-side and air-side. Conversely, a higher inlet steam quality or higher inlet steam mass flow rate resulted in higher heat rejection rates, with

the mass flow rate having a stronger correlation with the total heat rejection rate. Hence, the total heat rejection rate was strongly influenced by steam-side properties. Higher steam qualities and steam mass flow rates required more steam to be condensed by the ACC system, resulting in larger heat rejection rates.

Figure 51 shows the ambient air temperature and backpressure distributions for dataset 1 and 2 based on the number of streets operating. Notably, switching off streets allowed for the backpressure to be maintained at lower ambient air temperatures. This was observed when comparing the lowest ambient air temperatures solved between the datasets. The sample with the lowest ambient air temperature that was solved by the thermofluid model with all streets operating was $11.8°C$, compared to $7.9°C$ and $5.0°C$ for one street and two streets switched off respectively. Switching off streets resulted in higher backpressures, as the ACC system had a lower cooling capacity due to non-operating streets which required a higher backpressure to condense all inlet steam. This allowed for lower ambient air temperatures to be considered while maintaining the backpressure above $7\ kPa$ as seen in the bottom graph of Figure 51.

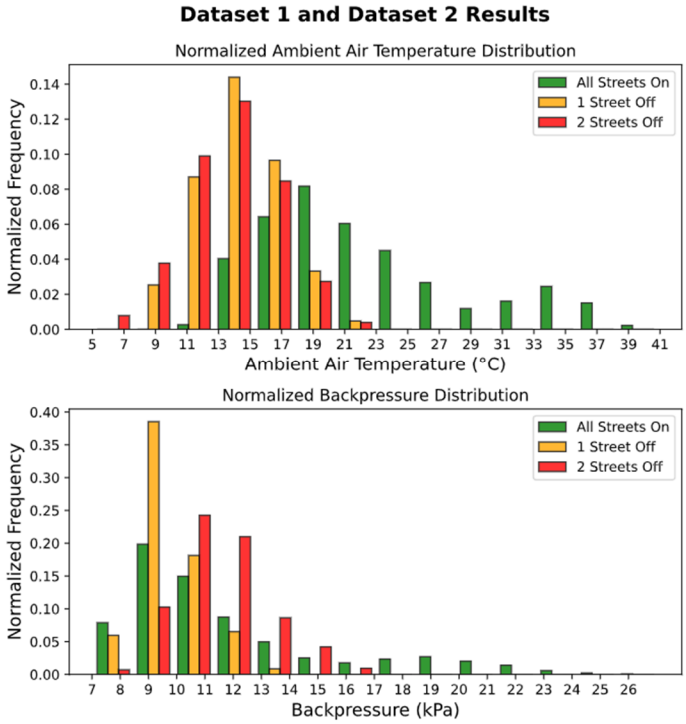


Figure 51: Ambient air temperature and backpressure distributions for datasets 1 and 2

Figure 52 shows the total air volume flow rate results from dataset 3 against the wind angle and fan speed reduction. For the wind angle graph on the left of Figure 52, for each wind angle, seven different clusters of air volume flow rates can be observed. These directly correlate with the seven

increments of fan speed reductions used. The air volume flow rate for a given wind angle would, therefore, decrease sequentially with higher fan speed reductions, resulting in the clusters as mentioned above.

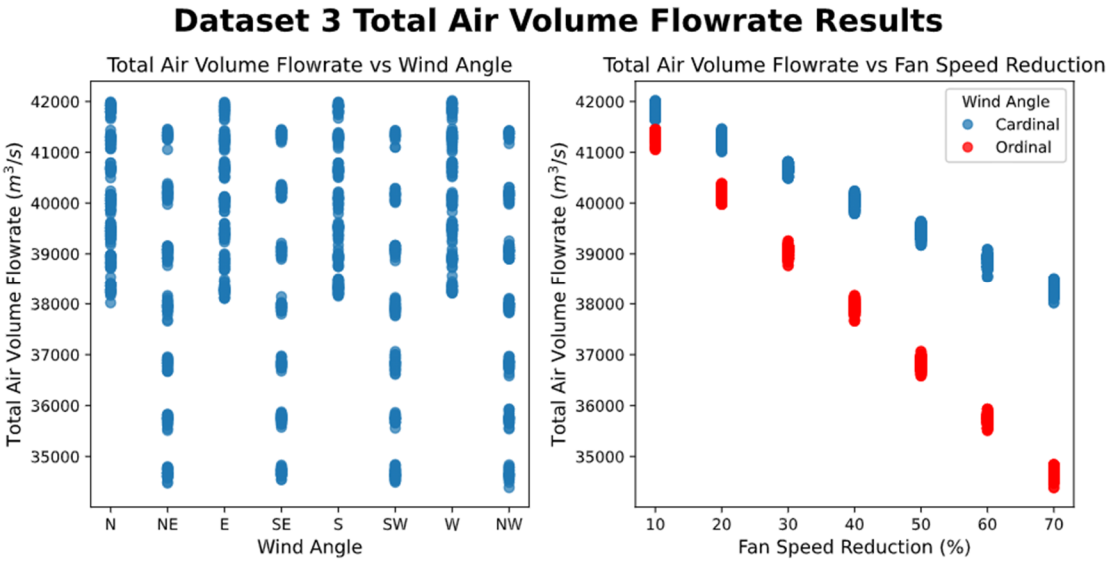


Figure 52: Total air volume flow rate results against wind angle and fan speed reduction from dataset 3

Notably, for cardinal directions (N, E, S, W), the air volume flow rate decreased less compared to ordinal wind directions (NE, SE, SW, NW). Comparing the air volume flow rate difference from 10% fan speed reduction to 70% fan speed reduction, an approximate decrease of 4000 m^3/s compared to 8000 m^3/s was observed for cardinal and ordinal wind angles respectively. Moreover, the clusters on the left graph of Figure 52 (for each fan speed reduction increment) were more spaced out for ordinal wind angles compared to cardinal wind angles. This was because ordinal wind angles resulted in both a street as well as a column of cells being affected by wind, compared to just one of the two for a cardinal direction. Consequently, the air volume flow rate reduction would be larger as more cells were influenced by wind effects. Considering the graph on the right of Figure 52, with increasing fan speed reduction, the air volume flow rate difference between cardinal and ordinal wind directions increased. As more cells were influenced by the wind for ordinal wind angles, an increased fan speed reduction would have a more significant effect on the total air volume flow rate compared to a cardinal wind angle.

As mentioned previously, the results obtained from the thermofluid model in tandem with the provided inputs from the DOEs were used to form the final combined dataset for the regression MLP network. The same inputs were also used for the binary classifier MLP network and were marked with the output as solvable. The inputs that were unsolvable by the thermofluid network

model were also used and were marked as non-solvable. Artificial inputs were generated to supplement the non-solvable class to obtain a balanced dataset with a similar amount of solvable and non-solvable output labels. These inputs were generated based on the outlying ranges of the input spaces shown for each dataset in Figure 44, Figure 46, and Figure 48, to avoid extrapolation within these outlying input spaces. These inputs were generated using specified increments for each of the three datasets. The dataset for the binary classifier contained 4034 solvable inputs and 2600 non-solvable inputs, as shown in Figure 53. There was a slight imbalance between the number of solvable and non-solvable inputs; however, as shown later, the binary classifier network was still able to perform relatively well. Figure 53 also shows the breakdown of solvable and non-solvable samples into different datasets. Majority of samples were from datasets 1 and 3, as these datasets covered much wider input spaces compared to dataset 2. The breakdown of the solvable inputs also constitutes the breakdown of the final combined dataset used for the regression MLP network.

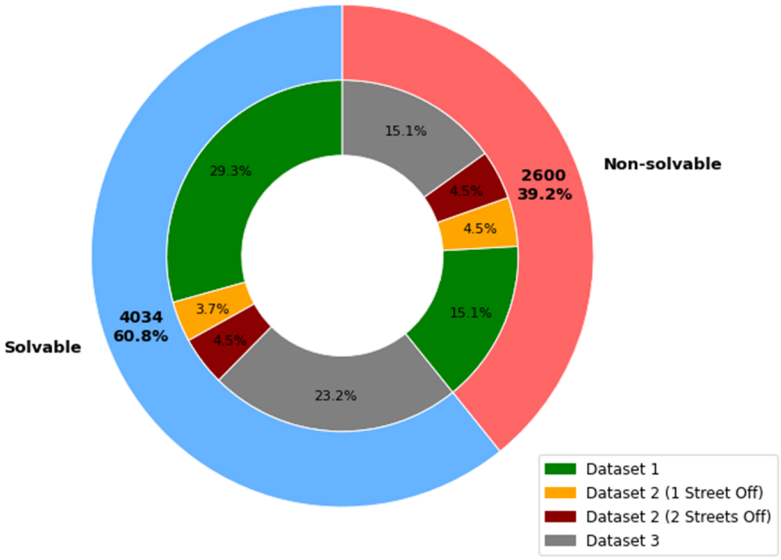


Figure 53: Breakdown of binary classifier dataset sample sizes

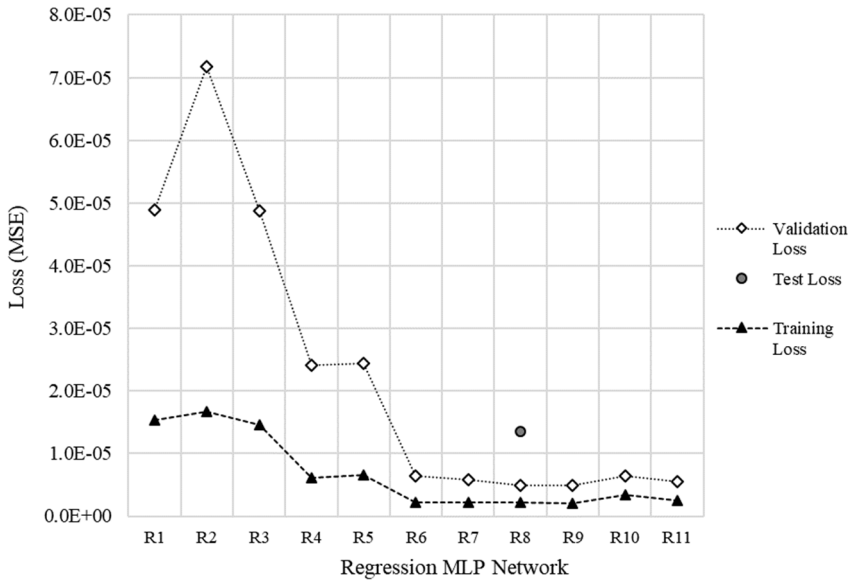
The developed datasets, including the regression dataset and binary classifier dataset, were used to train and test their respective MLP networks. The regression MLP network results and discussion was presented, followed by the binary classifier MLP network.

5.2.2 Regression MLP Network

The regression MLP network was formed through a hyperparameter search process consisting of 11 different MLP networks with various hyperparameter values, as shown in Table 26. The final model was selected based on the lowest validation loss, and the performance of the final MLP network

was quantified through the test set performance. The training set consisted of 3227 samples, while the test set had 807 samples, which was an 80/20 split, respectively.

Figure 54 shows the average training and validation loss for each of the 11 regression MLP networks. The final MLP network selected was R8 (see Table 26). With the hyperparameter tuning process, the training and validation losses fluctuated as the hyperparameters were altered. Generally, a reduction in training loss resulted in a reduction in validation losses. The losses shown in Figure 54 were calculated using the transformed network values through the min-max scaler.



Hyperparameter	R8
Batch size	64
Learning rate	1.0E-4
Epochs	500
Hidden layers	5
Neurons per hidden layer	4096

Figure 54: Validation and training losses for 11 regression MLP networks in hyperparameter search (left) and final hyperparameters for selected regression MLP network (right)

As the hyperparameters were manually tuned, training and validation losses decreased to an optimum, resulting in the selection of network R8. Starting from R1, which was a two-layer network with 4096 neurons per hidden layer, to R2, which was a two-layer network with 2048 neurons per hidden layer, the validation loss increased. This was due to the reduced number of hidden neurons per hidden layer, which reduced the fitting capacity of the network. Additionally, the batch size from R1 to R2 was doubled, requiring more epochs to arrive at the minimum validation loss, with 990 epochs for R2 compared to 499 for R1. R3 had similar performance on the validation set compared to R1, as only the batch size was doubled. However, for R4, the batch size was halved, and the learning rate increased to 1E-3, which nearly halved the validation loss. For R5, the neurons per hidden layer were increased to 8000, which did not have much effect on validation performance.

Consequently, in R6, the number of hidden layers was increased from 2 to 3. This resulted in a further decrease in validation losses. The number of hidden layers was then increased to 4 in R7 and

finally 5 in R8, resulting in the final model architecture. R9 investigated a batch size of 32 as opposed to 64 in R8, but this increased training time with no reduction in training and validation losses. R10 had an increase in neurons per hidden layer, which showed an increase in both validation and training loss. Additionally, R11 had six hidden layers, which also had an increase in training and validation loss. R10 and R11 were trained for 425 and 500 epochs, respectively. With early-stopping used for regularisation, the increase in training loss suggested that these networks were underfitting as they required more training time compared to R8 due to the additional trainable parameters. R10 and R11 would therefore have benefitted from more epochs while training. Nonetheless, with the presented parameters from Table 26, R8 was selected with the lowest validation loss.

The selected network architecture and hyperparameters for R8 is shown on the right of Figure 54. The average scaled test set loss was 1.34E-05. If inversed using the min-max scaler, the actual average loss for all output parameters was 4.255. The absolute and relative error analysis on the test set performance for each output parameter is presented in Table 28 in the form of average, maximum, and minimum errors between the predicted value from the MLP network and the actual value from the test dataset. The C and M abbreviations were used for condenser and mixed cell, respectively.

Table 28: Test set performance (absolute and relative errors) for each output parameter from MLP network R8

	Backpressure Error (*kPa*)	Total Heat Rejection Rate C Error (*MW*)	Total Heat Rejection Rate M Error (*MW*)	Total Air Mass Flow Rate C Error (*kg / s*)	Total Air Mass Flow Rate M Error (*kg / s*)	Total Fan Motor Power C Error (*kW*)	Total Fan Motor Power M Error (*kW*)
Average	0.031	0.363	0.197	6.360	2.660	2.437	0.702
Maximum	0.382	33.200	12.382	67.215	20.441	48.822	6.204
Minimum	1.34E-04	3.86E-04	2.20E-04	7.85E-04	6.02E-03	3.85E-03	9.22E-04
Avg (%)	0.291	0.053	0.094	0.022	0.028	0.029	0.026
Max (%)	4.886	4.535	5.553	0.234	0.267	0.578	0.211
Min (%)	1.19E-03	5.79E-05	9.67E-05	2.62E-06	7.92E-05	5.30E-05	3.14E-05

From Table 28, the selected regression MLP network performed relatively well, with an average error below 0.3% for all seven predicted parameters. The maximum error for the backpressure as

well as the total condenser and mixed cell heat rejection rate was relatively higher, under 5.6%. This was mainly due to a single outlier prediction, as shown later. On the other hand, the minimum relative error was relatively low, under 0.0012%.

Figure 55 shows the actual values from the dataset plotted against the predictions from the regression MLP network on the test set. A diagonal line indicated how well the predictions matched the actual values from the dataset. For all parameters, most predictions were on the diagonal line, which indicated that predictions matched actual values relatively well. There was a single outlier prediction, highlighted in red circles for the total heat rejection rate parameters. This outlier increased the maximum error, shown in Table 28, but was limited to a single sample.

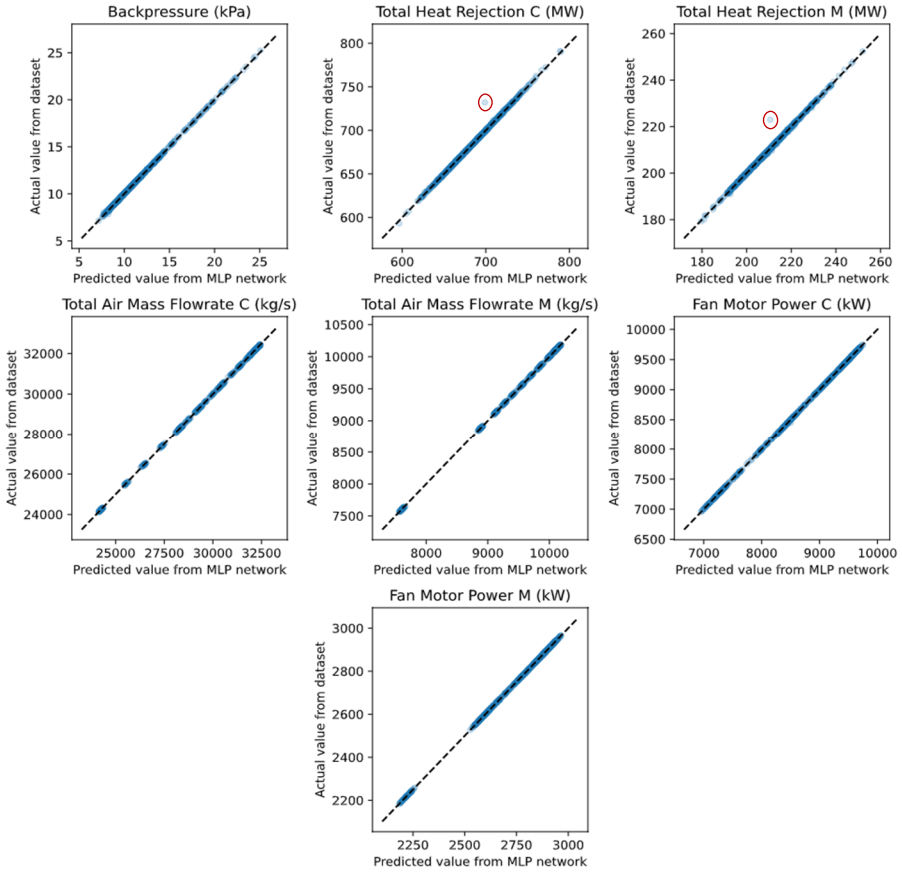


Figure 55: Actual value from dataset vs predicted value from regression MLP network for test set

Figure 56 shows the relative error distributions (histograms) between the predictions from the regression MLP network and actual dataset. For the total air mass flow rate and total fan power parameters, all errors were below 0.4%. For the backpressure, most samples were below 1%, with two outliers at 4% and 5%. Similarly, for the total heat rejection rate parameters, most of the errors

were below 1%, with an outlier at 5% for both condenser and mixed cell predictions, corresponding with the outlier sample shown in Figure 55 in the red circle.

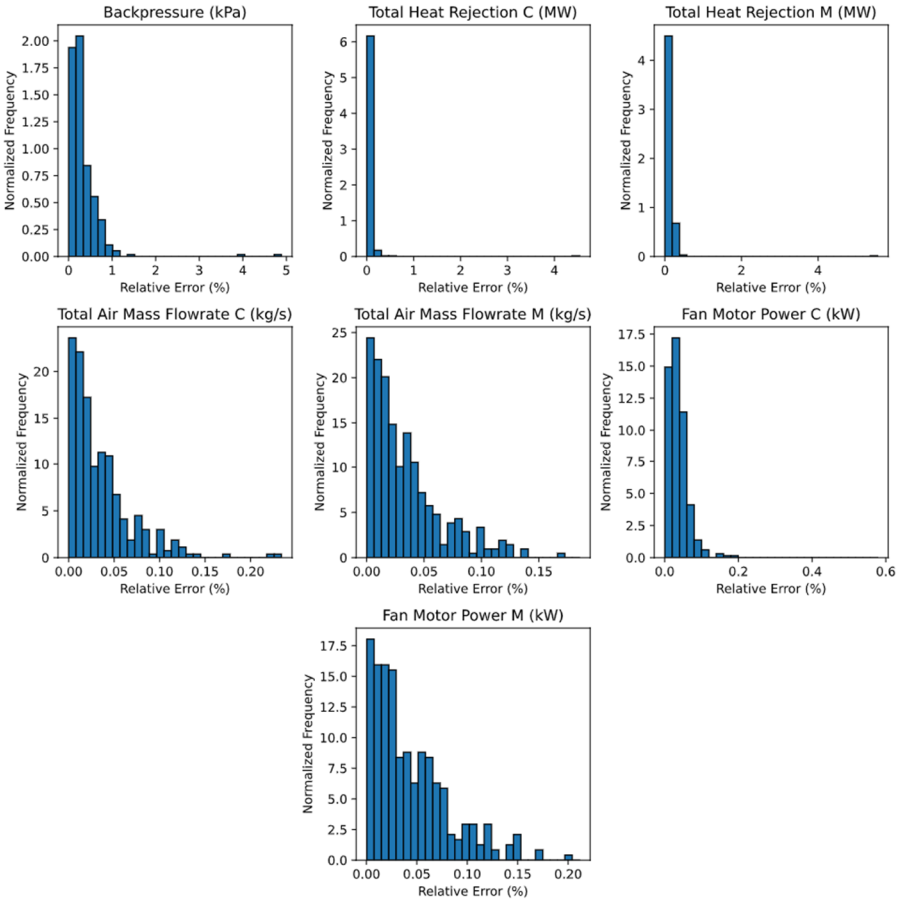


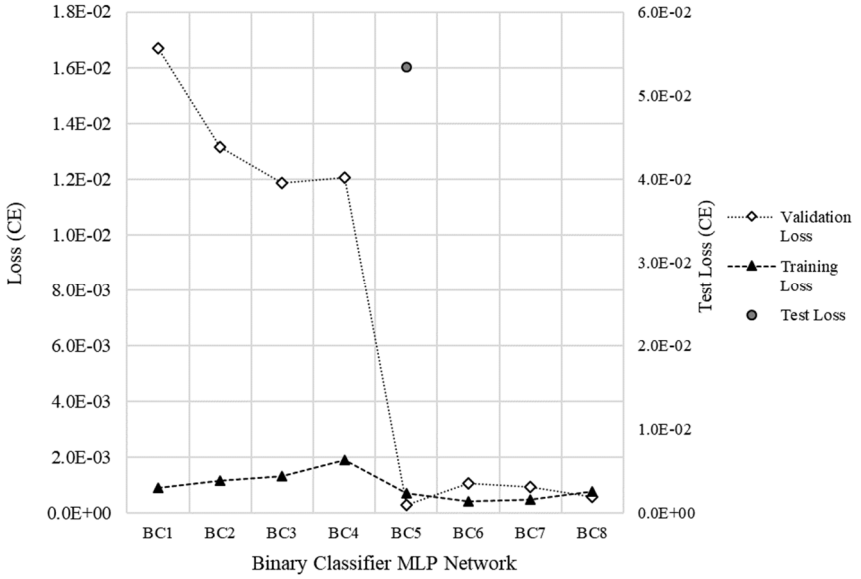
Figure 56: Test set relative error distribution for regression MLP network

From Figure 55, Figure 56, and Table 28, the regression MLP network was able to predict ACC performance with a 0.3% average relative error compared to the thermofluid network model. Therefore, the regression MLP network successfully acted as a data-driven surrogate model for the thermofluid network model specifically for the operating ranges of the datasets used to develop the data-driven surrogate model. The thermofluid network would still be required to solve for ambient and operating conditions that lay outside of the datasets used in this work, which would be flagged by the binary classifier. For inputs marked as solvable by the binary classifier, the regression MLP network would then be used to provide predictions within fractions of a second, as opposed to the thermofluid network model which took around 20-40 minutes to solve one steady-state simulation.

5.2.3 Binary Classifier MLP Network

The binary classifier MLP network underwent the same hyperparameter search process as the regression MLP network. Eight different MLP networks with various hyperparameters were used to find the optimal network architecture and hyperparameters for the binary classifier, shown in Table 27. Similarly, the validation loss was used to select the best performing network. The training set consisted of 5307 samples, while the test had 1294 samples, which was an 80/20 split, respectively.

Figure 57 shows the average training and validation losses for each of the eight binary classifier MLP networks used during the hyperparameter search. The final binary classifier MLP network selected was BC5, with the selected architecture and hyperparameters shown on the right of Figure 57. Consequently, BC5 had an associated test loss, evaluating the performance on the unseen test set.



Hyperparameter	BC5
Batch size	64
Learning rate	1.0E-3
Epochs	81
Hidden layers	3
Neurons per hidden layer	512

Figure 57: Validation and training losses for eight binary classifier MLP networks in hyperparameter search (left) and final hyperparameters for selected binary classifier MLP network (right)

From BC1 to BC2, the number of neurons per hidden layer was doubled from 512 to 1024, with two hidden layers in both BC1 and BC2, which resulted in a decrease in validation loss. BC3 had three hidden layers compared to two in BC2, which resulted in another slight decrease in validation loss. BC4 had a lower learning rate, 1E-4, compared to 1E-3 for the previous networks; however, this resulted in a small increase in the validation loss and the training loss. BC5 reverted to 512 neurons per hidden layer, but with three hidden layers. The learning rate was also increased back to 1E-3. This configuration had the lowest validation losses out of all the binary classifier network configurations used, indicating that the previous networks may have been overfitting. BC6 used an

additional hidden layer, but this resulted in a slight increase in validation loss. BC7 had three hidden layers and half the number of neurons per hidden layer compared to BC5, 256 compared to 512. BC8 had half the batch size compared to BC5; however, the rest of the hyperparameters were the same as BC5. Both BC7 and BC8 had higher validation losses compared to BC5 and were consequently not selected for the final model.

The selected binary classifier, BC5, had an associated test loss of 0.0533 and classification accuracy of 99.85%. To further investigate test set performance for the binary classifier, a confusion matrix was used in Figure 58, which shows how many samples were correctly labelled by the binary classifier. The selected binary classifier MLP network correctly labelled 480 non-solvable samples and 812 solvable samples. Only two samples were false-negatives, where the binary classifier labelled the samples as non-solvable when they were solvable. No samples were marked as a false-positive, which would be classified as being solvable by the binary classifier but were not solvable.

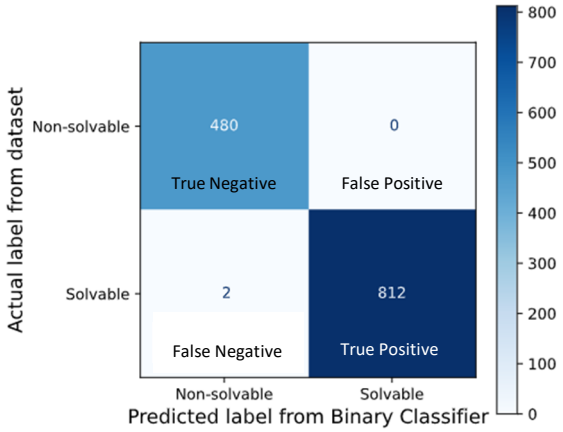


Figure 58: Confusion matrix for binary classifier MLP network on test set

Additional classification metrics include the precision, recall, and F1 score. Precision is the ratio of true positives to all marked positives, which is the classifier's ability not to produce false positives. Recall quantifies the ability of the classifier to find all positive samples, while the F1 score is a weighted average of both the recall and precision scores. All scores ranged from 0-1, where 1 is the best score. The selected binary classifier MLP network had a precision of 1.0, a recall of 0.998, and an F1 score of 0.999, which indicated the classifier could correctly label samples.

5.2.4 Validation to site-data

The developed data-driven surrogate model was validated using another set of site-data spanning over 10 days in 10-minute intervals (different to the design data used for validating the 1D thermofluid network model). The site-data included all required inputs to the model and only the

backpressure as the single output. Consequently, validation was only conducted on the backpressure predictions provided by the data-driven surrogate model. Additionally, the data-driven surrogate model could only account for cells switched off in 8-cell-increments (one street), while the site-data had instances where cells were switched off in finer increments (3 or 4 cells for example). To account for this, a scale to adjust the backpressure prediction was developed before predicting the backpressure using site-data. The differences in backpressure from operating 64 cells, 56 cells, and 48 cells were found at $22°C$ (highest temperature used in dataset 2) and design conditions. The backpressures at these operating fan increments were generated using the trained data-driven surrogate model. Accordingly, a 2^{nd}-order polynomial was fitted, which allowed for the backpressure to be scaled based on the number of cells operating. Figure 59 shows the change in predicted backpressure based on the number of fan cells operating, with the corresponding fitted polynomial function.

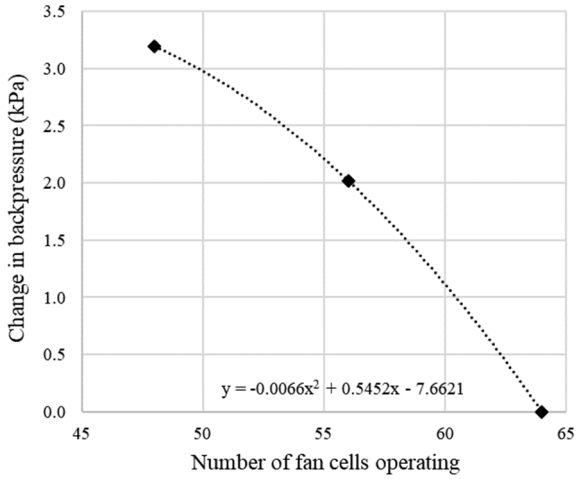


Figure 59: Change in backpressure prediction based on number of fan cells operating

Two sets of site-data were used for validation for two different ACC units, both with the same design as the case study ACC system. The average wind speed for the site-data was approximately $2.6\ m/s$, which was relatively low, resembling design conditions. There were 1096 samples processed in site-dataset 1, and 871 samples processed in site-dataset 2.

Figure 60 shows the ambient air temperature, inlet steam mass flow rate, recorded backpressures, and predicted backpressures for site dataset 1. Figure 61 shows the same for site dataset 2. Only samples that had backpressure predictions were plotted. The site-data included several samples outside the scope of the generated datasets which were consequently removed. For both site-datasets, the backpressure predictions provided by the model tracked the fluctuations in recorded backpressure, particularly the sharp rises in backpressure for site dataset 2. Figure 62 shows the

relative error histograms between recorded and predicted backpressures for both site datasets. For site dataset 1, 93.5% of samples were within ±6% error, while 95.9% of samples were within ±6% error for site dataset 2.

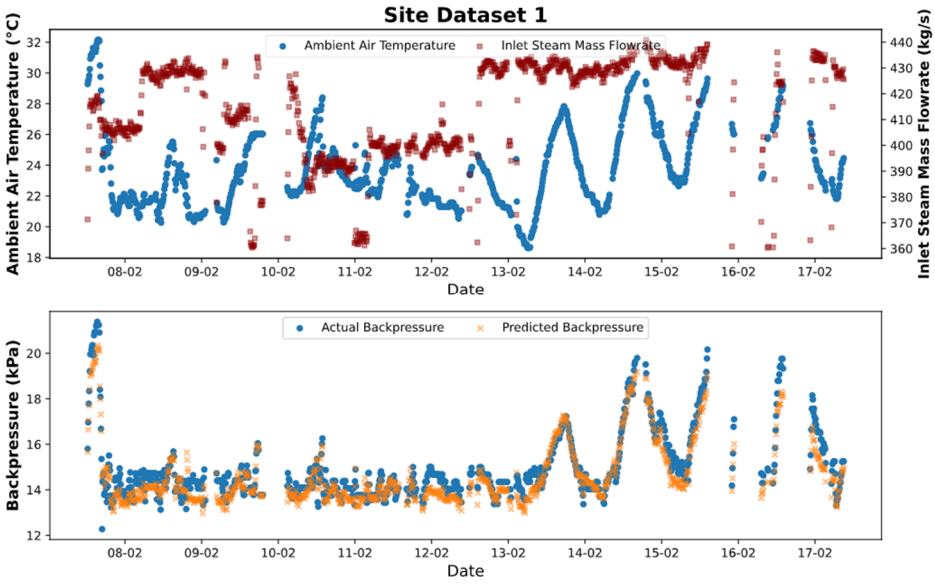


Figure 60: Site Dataset 1; Top: Ambient air temperatures and inlet steam mass flow rates from site-data; Bottom: Actual and predicted backpressures

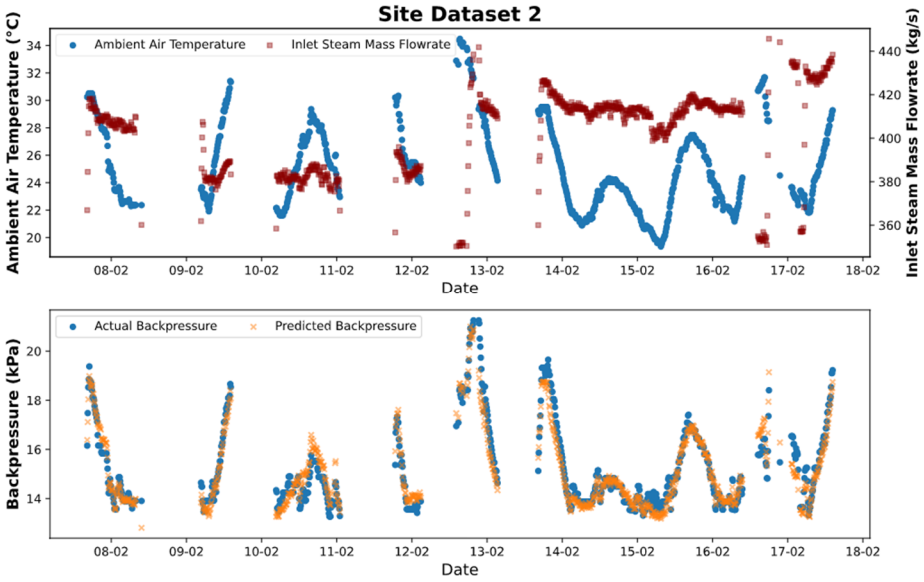


Figure 61: Site Dataset 2; Top: Ambient air temperatures and inlet steam mass flow rates; Bottom: Actual and predicted backpressures

The drawback to the data-driven surrogate model was the scope of the generated datasets, particularly at low unit loads. This is not a major drawback seeing as plant operators are typically

more concerned with condenser performance at high generating loads. However, the scope of the training dataset can be addressed in future work by generating more data with a wider scope of operating conditions and re-training the regression and binary classifier networks with a new hyperparameter search.

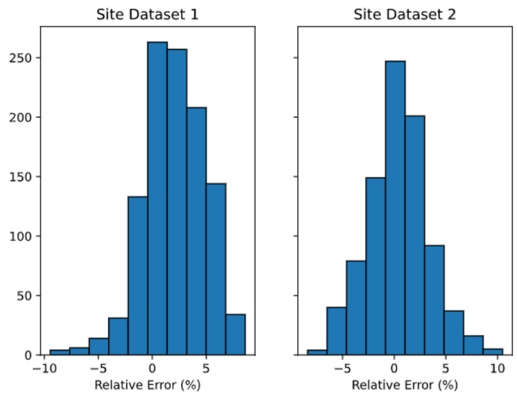


Figure 62: Relative error histograms between actual and predicted backpressures

5.2.5 Forecasting tool using web-application prototype

The developed binary classifier and regression MLP networks were then hosted together to form a condition monitoring platform through the proposed web-based forecasting tool. A Flask API [63] was used to host the ML models alongside HTML pages for user-interaction. Detailed instructions on how to use the web-app can be found in Appendix D, and the forecasting tool can be accessed as a deployed AWS Elastic Beanstalk web-application via https://accwebtool.af-south-1.elasticbeanstalk.com/, using the username User and the password RAH_Masters_2020. The forecasting tool provided weather forecast inputs to the data-driven surrogate model, which, in turn, provided ACC performance predictions. Figure 63 shows the ACC performance predictions based on a five-day weather forecast. Full unit load steam conditions were assumed, with an inlet steam quality of 0.9245 and an inlet steam mass flow rate of $422\ kg/s$. The forecast tool plots each output parameter from the regression MLP network, in addition to total heat rejection rates, total air mass flow rates, and total fan motor powers. The forecasted weather conditions were passed to the binary classifier, which then passed all samples that were solvable to the regression MLP network for predictions. Non-solvable forecasted inputs would not be passed to the regression MLP network and were therefore not plotted.

The developed condition monitoring platform, through the web-application prototype and forecasting tool, demonstrated the practicality of the data-driven surrogate model, and how these models could potentially be deployed on-site. The condition monitoring platform could potentially be run in parallel with a power plant and provide plant operators with performance predictions in

real-time. Notably, the current research acts as a stepping-stone in this respect, as the model still requires further validation concerning wind speeds, and the web-application requires further development.

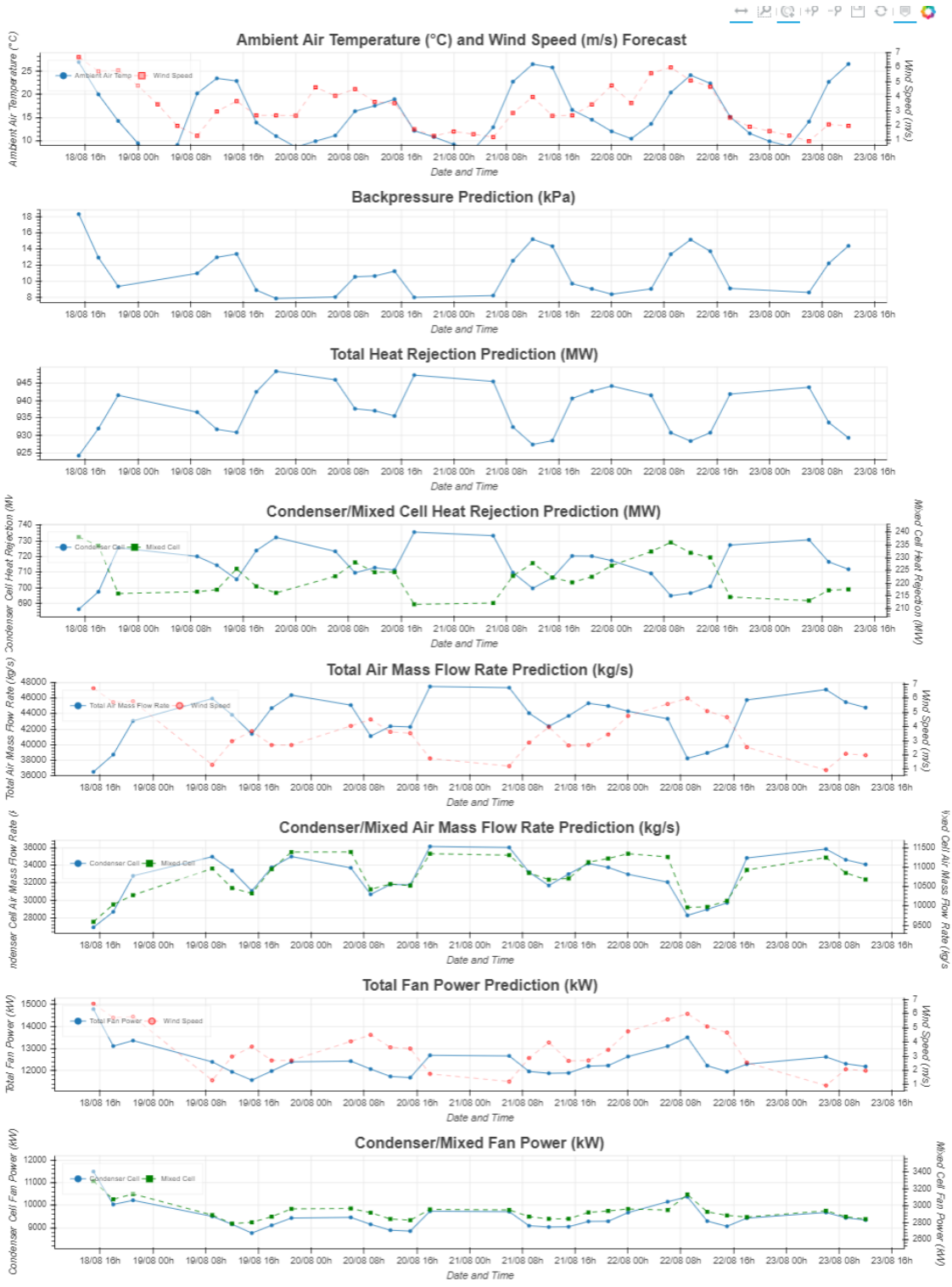


Figure 63: Forecasting tool predictions based on 5-day weather forecast

6. Conclusions and Recommendations

This chapter provides a summary of the main chapters of the report and concludes each chapter. A brief list of recommendations for future work is also presented thereafter.

The research focused on the development of a thermofluid network modelling methodology for double-row ACCs, as well as a condition monitoring platform using machine learning techniques to develop a data-driven surrogate model. Driven by the need to increase water and energy efficiency, dry cooling has seen a sharp rise in deployment. However, several barriers are impeding further adoption of dry cooling systems, namely, the reduced cooling efficiency and performance susceptibility to ambient conditions. ACCs are also deployed at several power stations by Eskom in South Africa. This demonstrated a need to be able to monitor ACC performance under varying ambient conditions to better plan electricity production. The project was idealised into two main sections, development of the thermofluid network model which solves the system on a cell-by-cell basis, and data-driven surrogate modelling. As an additional research output, a web-application prototype was developed to demonstrate the utility of data-driven surrogate models and how such models could be incorporated on-site.

A 1-D thermofluid network modelling approach was adopted using Flownex® SE to model a utility-scale ACC system at a power plant in South Africa. ACC modelling typically used computationally expensive CFD models; however, this approach is unfeasible for solving an entire ACC system which incorporates air-side and steam-side phenomena. A 1-D thermofluid network modelling approach sufficiently captured both cell-by-cell and system-level parameters at a compromise of a simplified air-side accuracy. The developed modelling methodology was based on the approach and correlations developed by Kröger [5] for condenser cells and modified for mixed cells which included dephlegmator heat exchangers. The vapour distribution duct flow into individual streets was modelled, along with the inlet header losses, momentum, frictional, and gravitational losses in the heat exchanger tube bundles, as well as the outlet header losses. The air-side was solved through a series of interconnected flow conduits, including a fan, based on the modified draft equation from Kröger [5] which incorporated a kinetic recovery factor. An iterative solution methodology was used to find a steady-state backpressure based on provided ambient and inlet steam conditions.

The developed steady-state 1-D thermofluid network model was then validated with site data at $27.21°C$ with a maximum relative error under 0.7%. The kinetic energy recovery factor, K_{rec}, was adjusted to 0.310. The 1-D thermofluid network model was also compared to the existing lumped approach based on Kröger [5], which had $K_{rec} = 0.255$. The existing lumped approach was solved for a single condenser cell and a mixed cell, which was then linearly scaled up to represent the

overall ACC system. Two investigations were conducted, the first under varying ambient air temperatures, and the second under hot air recirculation and wind effects.

For the varying ambient air temperature study from $25°C$ to $35°C$, the backpressure rose by $7.79\ kPa$ while the heat rejection rate increased by $10.9\ MW$ to condense all inlet steam. These results were also in agreement with the results predicted by the existing lumped approach. However, for the developed model, the inhomogeneous vapour duct distribution resulted in an increase in vapour flow rate to cells at the beginning of a street compared to cells towards the rear of a street, resulting in a slight gradient in cell heat rejection rates along the length of the streets. The existing lumped approach was unable to model the vapour duct distribution effects as only two cells within a street would be modelled to represent the entire system. For the developed model, the pressure losses in individual heat exchanger rows were quantified per cell. A reduction in heat exchanger tube pressure drops was observed at higher ambient air temperatures, due to the increased steam density with higher backpressures. Heat rejection rates for condenser heat exchangers decreased across the temperature range. In contrast, dephlegmator heat exchangers exhibited an initial increase in heat rejection rates, due to higher inlet vapour pressures and vapour flow rates. The average thermal resistances, as well as overall thermal heat transfer coefficient, were quantified on a cell-by-cell basis. Condensate heat transfer coefficients increased slightly over the temperature range due to an increase in vapour backpressure, while air heat transfer coefficients decreased.

For the hot air recirculation and wind effects study, the developed model was compared to the existing lumped approach. These effects were incorporated in the existing lumped approach using a cell-weighted average for the affected ambient air temperature or fan speed. With a hot air recirculation weighting up to 60%, a backpressure increase of $3.00\ kPa$ and a total heat rejection rate decrease of $3.15\ MW$ was observed for the developed model. Similarly, the existing lumped approach provided predictions within 0.5% of the developed model. For the wind effects study, wind effects were incorporated through the incremented reduction of fan speeds, and in turn, air volume flow rates. Unlike the hot air recirculation study, there was a noticeable difference between the developed model predictions and the lumped approach predictions for both backpressure and total heat rejection rate. The developed model predicted more than double the backpressure increase compared to existing lumped approach. The developed model also predicted a decreasing total heat rejection rate with larger fan speed reductions, while the existing lumped approach predicted an increasing total heat rejection rate trend. The differences in prediction trends were attributed to the presence of both wind-affected cells and unaffected cells in the developed model, as opposed to the cell-weighted average method used in the existing lumped approach.

For off-design conditions, such as hot air recirculation and wind effects, there was no useable site-data for validation. Consequently, the developed model was not validated for off-design modelling and would require future work for validation. A CFD model would also be a beneficial tool to develop a mapping between wind speed and an air volume flow rate reduction per cell; however, this was unavailable for the ACC system modelled in this work. Additional site-data could be used for further validation at design conditions. The thermofluid network model can be developed further and integrated into a co-simulation environment with a more robust air-side model using CFD, potentially allowing for detailed air-side flow to be incorporated and for the effects on steam-side performance to be quantified.

A data-driven surrogate model was created to improve the practicality and utility of the developed thermofluid network. The primary use of the data-driven surrogate model was to provide ACC performance predictions on a system level in an online fashion. Consequently, the data-driven surrogate model was based on a simplified version of the thermofluid network model that did not account for the vapour duct distribution to individual cells. The modelling process for the data-driven surrogate model was split into data generation and machine learning modelling through the regression and binary classifier MLP networks. Data generation was completed using inputs generated through Latin-hypercube sampling with three datasets covering a range of ambient and ACC operating conditions. The first dataset considered all the ACC streets operating without wind effects. The second dataset considered up to two streets switched off without wind effects. The third dataset accounted for wind effects with all ACC streets operating. The generated inputs were then passed to the thermofluid network model and overall ACC system results recorded. A total of 4700 input samples were generated, and 4034 samples were successfully solved by the thermofluid network model, which then formed the final dataset used to train the regression MLP network. A further 2600 augmented non-solvable inputs were generated for training the binary classifier MLP network.

With the datasets constructed from the data generation process, the regression and binary classifier MLP networks were developed. Both the regression and binary classifier MLP network used the feed-forward and backpropagation algorithms in conjunction with the Adam optimizer to adjust network parameters. Early stopping was used as the main form of regularisation for ML modelling. The ReLU activation function was used for all hidden layers. The output layer of the regression MLP network had a linear activation function. For the binary classifier MLP network, a sigmoid activation function was used to determine the probability of classifying an input sample as solvable. A manual hyperparameter search was used to find the optimal network architecture and hyperparameters for the regression and binary classifier MLP networks. The hyperparameters tuned included the batch size, learning rate, epochs, hidden layers, and the number of neurons per hidden layer.

Consequently, 11 different regression MLP networks and eight binary classifier MLP networks were constructed with various hyperparameters. Six-fold cross-validation was used to select the best performing network based on the lowest average validation loss. For both the regression and binary classifier MLP networks, an 80/20 split was used to form the training dataset and testing dataset respectively.

The final regression MLP network had five hidden layers and 4096 neurons per hidden layer. The associated average test loss for all output parameters was 4.255. The average relative error was 0.3% between network predictions and actual values from the test dataset. A single outlier prediction resulted in a maximum relative error below 6%; however, most predictions had a relative error below 1%. Consequently, the selected regression MLP network was able to generalise sufficiently on the test set. Generalisation was observed for samples from all three datasets considering various ambient conditions and ACC operating conditions. Furthermore, the data generation process was satisfactory as the regression MLP network was able to generalise adequately as observed with the test set. The final selected binary classifier MLP network had three hidden layers with 512 neurons per hidden layer. The associated test classification accuracy was 99.85%, which was sufficiently accurate. Only two samples were incorrectly classified as false negatives.

The binary classifier was able to adequately predict whether a provided set of input samples were within the scope of the datasets used to develop the data-driven surrogate model. While the regression MLP network was limited with regards to the range of ambient and ACC operating conditions considered in the dataset, the thermofluid network model could be used to generate more data considering an even broader scope. Additionally, the predictions provided with consideration of wind effects were unverified, as the thermofluid network model itself was not verified for off-design conditions including wind effects. With a CFD model and more site-data to verify predictions considering wind effects, the utility of the thermofluid network model, and in turn, the data-driven surrogate model can potentially be improved. This would require the process of data generation and hyperparameter searches to be repeated, which needs to be considered for future work.

The developed data-driven surrogate model was hosted as a web-based forecasting tool prototype. The forecasting tool provided ACC performance predictions based on a 5-day weather forecast of the ACC system locality. Additionally, the forecasting tool also provided predictions for user-provided inputs. The web-application prototype demonstrated the utility and deployment of data-driven surrogate models and how such models could potentially be deployed on-site. Web-applications also have the benefit of being locally hosted and provide easy accessibility across a local network. Moreover, the required libraries were only required on the local computer hosting the

web-application, while users can access the web-app through a web-browser without installing any additional software. While the web-app development was intended to be a prototype, it requires further development to be deployed.

Recommendations are presented below, which could be considered to build upon the present research in future work:

1. Obtain more site-data to validate the thermofluid network model at design and off-design conditions, including wind effects. This would make predictions from the thermofluid network model more reliable, particularly for off-design conditions which have not been validated in the current research.
2. Develop a CFD model for the ACC system considered in this research, allowing for a robust mapping between wind speeds and air volume flow rate reductions. Additionally, the viability of co-simulation with the 1-D thermofluid network model and a CFD model could be explored, considering more complex air-side flow and the effects on steam-side performance.
3. Generate a larger dataset considering a more comprehensive range of ambient and ACC operating conditions. This could also include enough data samples to consider individual cells within the ACC system; however, this would require significantly more data samples. As a result, the data-driven surrogate model could provide predictions on a cell-by-cell basis.
4. Develop the web-application prototype further and deploy the web-app on a local server. If the thermofluid network model can be validated for off-design conditions, the data-driven surrogate model would need to be re-trained and deployed with a broader operating range. The web-app could then potentially be deployed on-site to investigate the forecasting tool's performance. An API could also be developed to allow for the web-app to be integrated into an existing software stack.

7. List of References

[1] UNFCCC, "The Paris Agreement | UNFCCC," *2018*. [Online]. Available: https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement. [Accessed: 28-May-2019].

[2] K-CEP, "Kigali Cooling Efficiency Program (K-CEP)." [Online]. Available: https://www.k-cep.org/. [Accessed: 28-May-2019].

[3] Y. Cengel and M. Boles, *Thermodynamics: An Engineering Approach*, 9th Editio. McGraw-Hill, 2019.

[4] W. Wurtz, "Air-cooled condensers eliminate plant water use," 2008. [Online]. Available: https://www.powermag.com/air-cooled-condensers-eliminate-plant-water-use/?pagenum=1. [Accessed: 09-Apr-2019].

[5] D. G. Kröger, *Air-Cooled Heat Exchangers and Cooling Towers*. Stellenbosch University, 1998.

[6] Eskom, "Dry cooling technology," 2016. [Online]. Available: http://www.eskom.co.za/news/Pages/Feb4X.aspx. [Accessed: 21-May-2019].

[7] WEF, "The Global Risks Report," 2019.

[8] J. G. Bustamante, A. S. Rattner, and S. Garimella, "Achieving near-water-cooled power plant performance with air-cooled condensers," *Appl. Therm. Eng.*, vol. 105, pp. 362–371, 2016.

[9] Electric Power Research Institute (EPRI), "California Energy Commission: Comparison of Alternate Cooling Technologies for California Power Plants Economic, Environmental and Other Tradeoffs," California, 2002.

[10] S. Lennon, "Advances in Dry Cooling Deployed at South African Power Stations," 2011.

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations."

[12] A. O'Donovan and R. Grimes, "Pressure drop analysis of steam condensation in air-cooled circular tube bundles," *Appl. Therm. Eng.*, vol. 87, pp. 106–116, 2015.

[13] L. Yang, H. Tan, X. Du, and Y. Yang, "Thermal-flow characteristics of the new wave-finned flat tube bundles in air-cooled condensers," *Int. J. Therm. Sci.*, vol. 53, pp. 166–174, 2012.

[14] W. A. Davies and P. Hrnjak, "Thermo-hydraulic model for steam condensation in a large, inclined, flattened-tube air-cooled condenser," *Appl. Therm. Eng.*, vol. 149, pp. 745–756, 2019.

[15] J. R. Bredell, D. G. Kröger, and G. D. Thiart, "Numerical investigation of fan performance in a forced draft air-cooled steam condenser," *Appl. Therm. Eng.*, vol. 26, no. 8–9, pp. 846–852, Jun. 2006.

[16] P. J. Hotchkiss, C. J. Meyer, and T. W. Von Backström, "Numerical investigation into the effect of cross-flow on the performance of axial flow fans in forced draught air-cooled heat exchangers," *Appl. Therm. Eng.*, vol. 26, no. 2–3, pp. 200–208, Feb. 2006.

[17] W. H. Stinnes and T. W. Von Backström, "Effect of cross-flow on the performance of air-cooled heat exchanger fans," *Appl. Therm. Eng.*, vol. 22, no. 12, pp. 1403–1415, Aug. 2002.

[18] F. G. Louw, T. W. Von Backström, and S. J. Van Der Spuy, "Lift and drag characteristics of an air-cooled heat exchanger axial flow fan," *J. Fluids Eng. Trans. ASME*, vol. 137, no. 8, Aug. 2015.

[19] M. T. F. Owen, "A numerical investigation of air-cooled steam condenser performance under windy conditions," University of Stellenbosch, 2010.

[20] R. A. Engelbrecht, "Numerical Investigation of Fan Performance in a Forced Draft Air-cooled Condenser," University of Stellenbosch, 2018.

[21] M. B. Wilkinson, F. G. Louw, S. J. Van Der Spuy, and T. W. Von Backström, "A comparison of actuator disc models for axial flow fans in large air-cooled heat exchangers," in *Proceedings of the ASME Turbo Expo*, 2016, vol. 1.

[22] C. J. Meyer and D. G. Kröger, "Plenum chamber flow losses in forced draught air-cooled heat exchangers," *Appl. Therm. Eng.*, vol. 18, no. 9–10, pp. 875–893, 1998.

[23] R. A. Engelbrecht, C. J. Meyer, and J. Van Der Spuy, "Modeling Strategy for the Analysis of Forced Draft Air-Cooled Condensers Using Rotational Fan Models," *J. Therm. Sci. Eng. Appl.*, vol. 11, no. 5, Oct. 2019.

[24] L. J. Yang, X. Z. Du, and Y. P. Yang, "Space characteristics of the thermal performance for air-cooled condensers at ambient winds," *Int. J. Heat Mass Transf.*, vol. 54, pp. 3109–3119, 2011.

[25] L. Chen, L. Yang, X. Du, and Y. Yang, "A novel layout of air-cooled condensers to improve thermo-flow performances," *Appl. Therm. Eng.*, vol. 165, pp. 244–259, 2016.

[26] K. Hooman and H. Gurgenci, "Porous Medium Modeling of Air-Cooled Condensers," *Transp Porous Med*, vol. 84, pp. 257–273, 2010.

[27] H. Deng and J. Liu, "Performance prediction of finned air-cooled condenser using a conjugate heat-transfer model," *Appl. Therm. Eng.*, vol. 150, pp. 386–397, 2019.

[28] K. Kekaula, Y. Chen, T. Ma, and Q.-W. Wang, "Numerical investigation of condensation in inclined tube air-cooled condensers," *Appl. Therm. Eng.*, vol. 118, pp. 418–429, 2017.

[29] A. J. Mahvi, A. S. Rattner, J. Lin, and S. Garimella, "Challenges in predicting steam-side pressure drop and heat transfer in air-cooled power plant condensers," *Appl. Therm. Eng.*, vol. 133, pp. 396–406, 2018.

[30] J. Moore, R. Grimes, E. Walsh, and A. O'donovan, "Modelling the thermodynamic performance of a concentrated solar power plant with a novel modular air-cooled condenser," *Energy*, vol. 69, pp. 378–391, 2014.

[31] M. Owen and D. G. Kröger, "A numerical investigation of vapor flow in large air-cooled condensers," *Appl. Therm. Eng.*, vol. 127, pp. 157–164, 2017.

[32] J. Lin, A. J. Mahvi, T. S. Kunke, and S. Garimella, "Improving Air-Side Heat Transfer Performance in Air-Cooled Power Plant Condensers," *Appl. Therm. Eng.*, p. 114913, Jan. 2020.

[33] J. A. Heyns, "Performance characteristics of an air-cooled steam condenser incorporating a hybrid (dry/wet) dephlegmator," Stellenbosch : Stellenbosch University, 2008.

[34] V. Gadhamshetty, N. Nirmalakhandan, M. Myint, and C. Ricketts, "Improving Air-Cooled Condenser Performance in Combined Cycle Power Plants," *J. Energy Eng.*, vol. 132, no. 2, pp. 81–88, 2006.

[35] L. Klimeš, J. Pospíšil, J. Štětina, and P. Kracík, “Semi-empirical balance-based computational model of air-cooled condensers with the A-frame layout,” *Energy*, vol. 182, pp. 1013–1027, 2019.

[36] L. Klimeš, J. Pospíšil, J. Štětina, and P. Kracík, “Semi-empirical balance-based computational model of air-cooled condensers with the A-frame layout,” *Energy*, vol. 182, pp. 1013–1027, Sep. 2019.

[37] L. Liang, W. Mao, and W. Sun, “A feasibility study of deep learning for predicting hemodynamics of human thoracic aorta,” *J. Biomech.*, vol. 99, 2020.

[38] H. Wu, X. Liu, W. An, S. Chen, and H. Lyu, “A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils,” *Comput. Fluids*, vol. 198, 2020.

[39] A. Warey, S. Kaushik, B. Khalighi, M. Cruse, and G. Venkatesan, “Data-driven prediction of vehicle cabin thermal comfort: using machine learning and high-fidelity simulation results,” *Int. J. Heat Mass Transf.*, vol. 148, 2020.

[40] Y. Shi, W. Zhong, X. Chen, A. B. Yu, and J. Li, “Combustion optimization of ultra supercritical boiler based on artificial intelligence,” *Energy*, vol. 170, pp. 804–817, 2019.

[41] M. Fast and T. Palmé, “Application of artificial neural networks to the condition monitoring and diagnosis of a combined heat and power plant,” *Energy*, vol. 35, pp. 1114–1120, 2009.

[42] R. Laubscher, “Time-series forecasting of coal-fired power plant reheater metal temperatures using encoder-decoder recurrent neural networks,” *Energy*, vol. 189, 2019.

[43] R. Dhanuskodi, R. Kaliappan, S. Suresh, N. Anantharaman, A. Arunagiri, and J. Krishnaiah, “Artificial Neural Networks model for predicting wall temperature of supercritical boilers,” 2015.

[44] H. Wang, W. Cai, and Y. Wang, “Modeling of a hybrid ejector air conditioning system using artificial neural networks.”

[45] X. Li *et al.*, “A data-driven model for the air-cooling condenser of thermal power plants based on data reconciliation and support vector regression,” *Appl. Therm. Eng.*, vol. 129, pp. 1496–1507, 2018.

[46] X. Du *et al.*, “Back pressure prediction of the direct air cooled power generating unit using the artificial neural network model,” *Appl. Therm. Eng.*, vol. 31, pp. 3009–3014, 2011.

[47] A. O’donovan, R. Grimes, E. . Walsh, J. Moore, and N. Reams, “Steam-Side Characterisation of a Modular Air-Cooled Condenser,” in *International Mechanical Engineering Congress & Exposition*, 2012.

[48] Flownex, “Flownex Theory Manual.” pp. 1–40, 2019.

[49] G. P. Greyvenstein, “An implicit method for the analysis of transient flows in pipe networks,” *Int. J. Numer. Methods Eng.*, vol. 53, no. 5, pp. 1127–1143, 2002.

[50] I. . Idelchik, *Handbook of Hydraulic Resistance*, 3rd ed. BHB, 2008.

[51] O. P. H. Augustyn, “Experimental and numerical analysis of axial flow fans,” Stellenbosch University, 2013.

[52] M. D. Mckay, R. J. Beckman, and W. J. Conover, “A Comparison of Three Methods for Selecting

Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 42, no. 1, p. 55, Feb. 2000.

[53] Z.-Z. Liu, W. Li, and M. Yang, "Two General Extension Algorithms of Latin Hypercube Sampling," *Math. Probl. Eng.*, vol. 2015, 2015.

[54] G. D. Wyss and K. H. Jorgensen, "A User's Guide to LHS: Sandia's Latin Hypercube Sampling Software," Feb. 1998.

[55] A. Lee, "Randomized Designs — pyDOE 0.3.6 documentation," *pyDOE*, 2014. [Online]. Available: https://pythonhosted.org/pyDOE/randomized.html#latin-hypercube. [Accessed: 15-Jul-2020].

[56] "MyForecast," *CustomWeather*, 2020. [Online]. Available: https://myforecast.com/. [Accessed: 15-Feb-2020].

[57] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd Editio. Sebastapol: O'Reilly, 2019.

[58] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[60] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[61] C. M. Bishop, *Pattern Recognition and Machine Learning*. Cambridge: Springer, 2006.

[62] "OpenWeatherMap," 2020. [Online]. Available: https://openweathermap.org/. [Accessed: 03-Aug-2020].

[63] "Flask Documentation (1.1.x)," *The Pallets Project*. [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/. [Accessed: 18-Aug-2020].

Appendix A. Dephlegmator Modelling and Results

This appendix details the other dephlegmator model developed during the research. A breakdown of the additional models, not described in the main text, is presented. A results comparison between the two dephlegmator models is presented, as well as with the existing lumped approach. It is important to note that these comparisons were performed for a different case study ACC, and the selected dephlegmator model was then used for the case study ACC presented in this work.

A.1 Dephlegmator Modelling

In total, two different dephlegmator models were developed. The first was the combined heat exchanger rows as used in the final model, which was presented earlier in Section 3.4.3. The second model, or alternative model, followed a similar method to the condenser heat exchangers, where each heat exchanger row was modelled individually. This model included more detail compared to the other models as it accounted for backflow between heat exchanger rows.

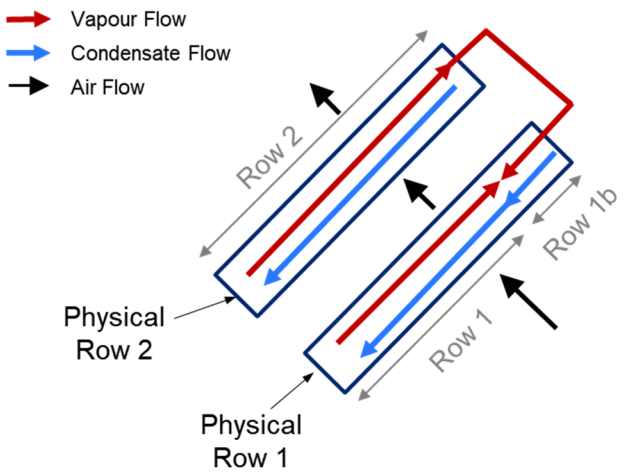


Figure 64: Second dephlegmator model heat exchanger row discretization

Figure 64 above shows the heat exchanger discretization for the second dephlegmator model. Heat exchanger row 1 was split to form a smaller row subsection at the top of the physical heat exchanger row called row 1b. This virtual split was done to accommodate any backflow occurring. Any steam that did not condense at the top of row 2 would then be inlet to row 1b. The lengths of each row segment were adjusted based on the outlet quality of the respective row.

Figure 65 shows the discretization used for row 1. The discretization was similar to the approach used in the final model; however, the length of row 1 was adjustable. This length was adjusted to

find the total condensing length of the heat exchanger. The reasoning behind a length adjustment was to find the total condensing length that resulted in condensation of all steam inlet to the row. The maximum length of row 1 was restricted by the physical length of the heat exchanger row, which was 10.4 *m* for dephlegmator heat exchangers. The difference in $L_{Row.1}$ and the maximum length, 10.4 *m* , was then used for the length of the subset row, row 1b.

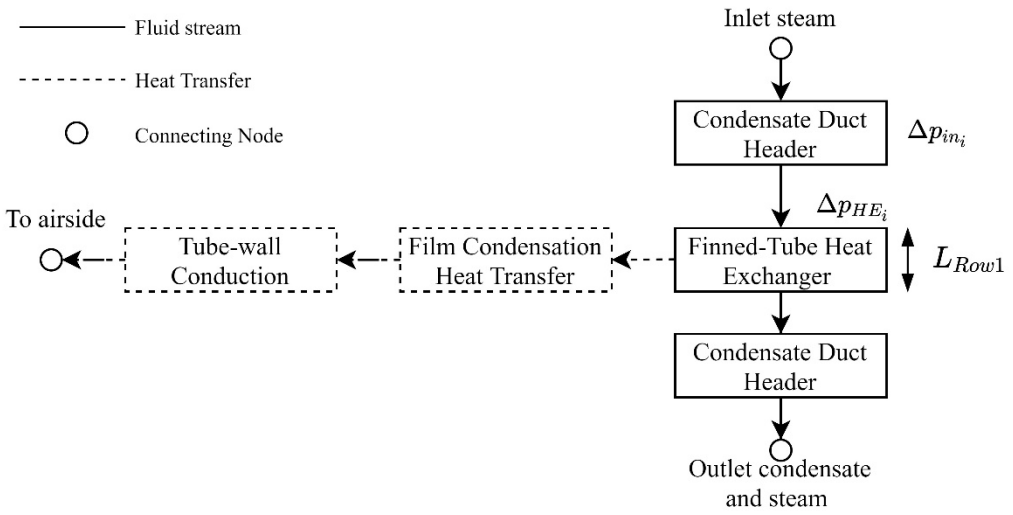


Figure 65: Row 1 discretization for second dephlegmator model

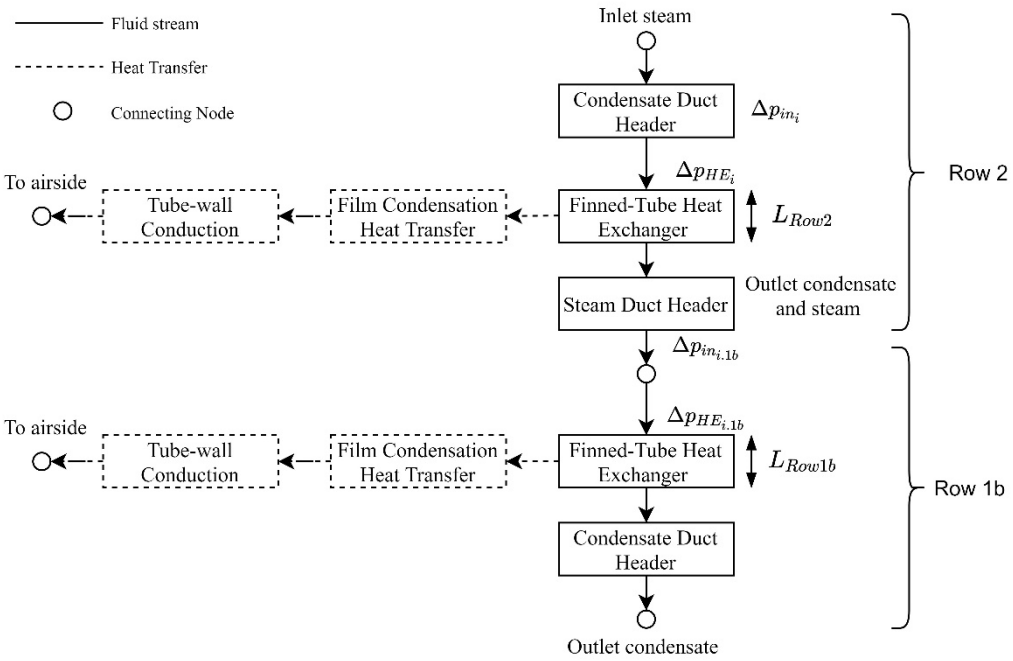


Figure 66: Row 2 and row 1b discretization for second dephlegmator model

Figure 66 shows the discretization for row 2 and row 1b. Row 2 and 1b were connected in series to account for backflow from row 2 into row 1. As a two-phase homogenous mixture assumption was used, both condensate and vapour streams at the outlet of row 2 would be inlet to row 1b. The physical characteristics of row 1b used, such as the number of tube bundles and flow areas, were from row 1 as row 1b was physically part of the first heat exchanger row. As mentioned in the previous paragraph, the length of the heat exchanger rows was variable. This was the same for row 2 as well as row 1b. If backflow occurred, row 2 would assume the full length of the physical heat exchanger row, $10.4\ m$. The length of row 1b would be the remaining length of the physical heat exchanger not used by row 1, $10.4\ m - L_{Row.1}$. If row 1 required the entire heat exchanger row length, $10.4\ m$, row 1b would have an assigned length of $0\ m$, and consequently, not have any effect on heat transfer.

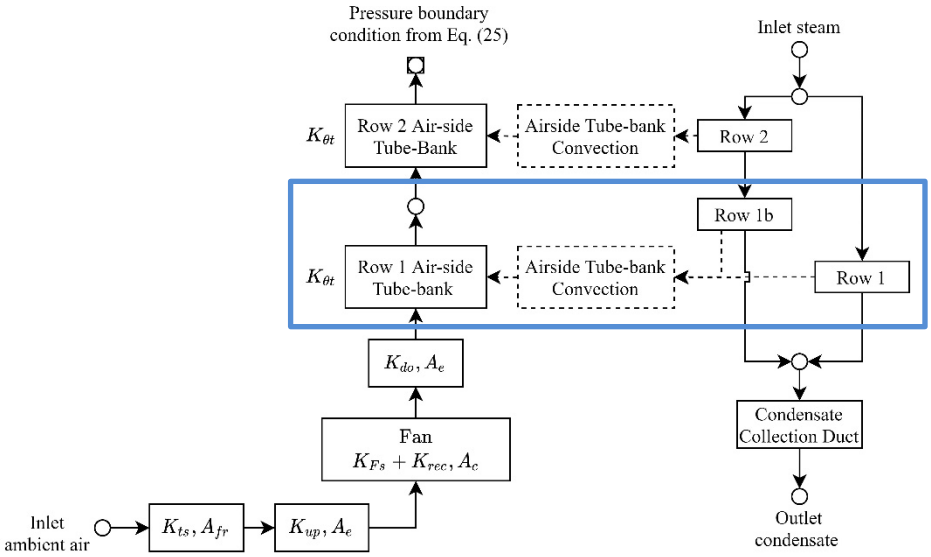


Figure 67: Dephlegmator discretization for second dephlegmator model

Figure 67 shows the discretization for the second dephlegmator model, including the steam-side and air-side. The notable difference to the first dephlegmator model discretization is the inclusion of row 1b on the steam-side. As row 1b had the physical characteristics of the first heat exchanger row, both row 1 and row 1b were linked to a single control volume for air-side flow, the row 1 air-side tube bank, as shown in the blue rectangle in Figure 67. This ensured a complete energy balance around the physical heat exchanger row on the air-side with the steam-side in row 1 and row 1b.

The solving methodology for the second dephlegmator differed to the adopted dephlegmator model. In addition to the variable lengths of the different heat exchanger rows, the consideration

of row 1b accounting for backflow changed the solution strategy from a fixed quality at the outlet of the dephlegmator to a variable quality at the outlet of row 1 and row 1b.

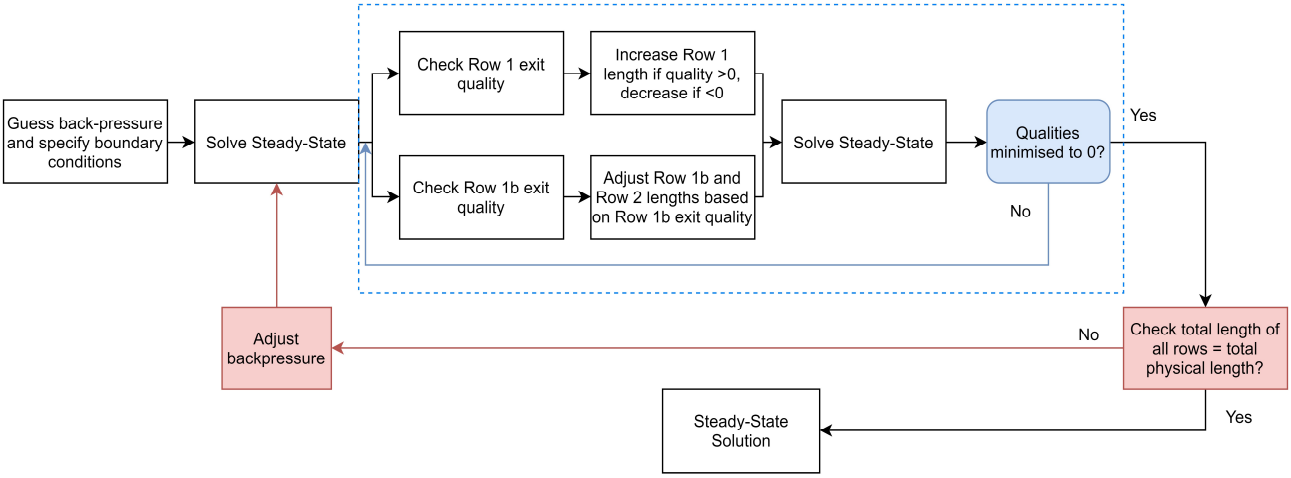


Figure 68: Solver methodology for second dephlegmator model

Figure 68 shows the solver methodology used for the second dephlegmator. The methodology consisted of two nested solver loops, as specified in blue and red, respectively. The first loop (blue) checked the row 1 and row 1b exit qualities to ensure they were minimised to 0. If the quality was >0, the respective row length would be increased, and if <0, the length decreased. Additionally, a constraint using the total combined length of the two heat exchanger rows, $10.4\ m + 10.4\ m = 20.8\ m$, was used to adjust the backpressure. If the adjusted total length was greater than the actual physical length of the heat exchanger rows, the backpressure would increase, and vice versa. A Newton-Raphson solver was used to calculate the adjusted lengths and backpressure within the nested loop.

A.2 Dephlegmator Results

The developed dephlegmator models were compared through based on a utility-scale 48 cell ACC system; however, only two streets were modelled for the dephlegmator model comparisons. The model results were also compared to the existing lumped approach, which was previously validated to site data.

Table 29 shows the backpressure and total heat rejection rate results for the two dephlegmator models as well as the existing lumped approach for two ACC streets at $15.6°C$. Notably, the final dephlegmator predicted slightly higher backpressures compared to the alternative model and the existing lumped approach. The difference in heat rejection rates between the two dephlegmator models and lumped approach was negligible, below 0.04%.

The over-estimate in backpressure from the final model as compared to the lumped approach could be due to the simplification of using the combined heat exchanger rows approach presented in Section 3.4.3, as compared to the more detailed approach presented in the alternative dephlegmator model.

Table 29: Dephlegmator model comparisons for two streets at 15.6°C

Parameter	Lumped Approach	Final Dephlegmator Model	Final Model Relative Error to Lumped Approach (%)	Alternative Dephlegmator Model	Alternative Model Relative Error to Lumped Approach (%)
Backpressure [kPa]	16.891	17.178	1.697	17.017	0.745
Total Heat Rejection Rate [MW]	225.277	225.339	0.0275	225.365	0.0389

The alternative dephlegmator model predicted backpressures with a slightly lower error relative to the lumped approach compared to the final model (0.745% compared to 1.697% respectively). However, the alternative model was much more computationally expensive, as well as unstable at times, depending on the initial backpressure and heat exchanger row lengths used. This was the main driving force in the selection of the final dephlegmator model used. The final model would solve within 15-20 minutes, compared to the alternative model taking 20-40 minutes. Additionally, the solution time for the alternative would vary greatly depending on the initial guesses for backpressure and non-physical lengths of the heat exchanger rows. When considering running the model under a wide range of ambient and operating conditions for generating data for the data-driven surrogate model, the final model was a clear choice in this regard, with a relatively small difference in backpressure predictions between the final model and alternative model.

Appendix B. Python API link to Flownex® SE

This notebook includes the relevant code used to pass input samples from spreadsheets to the developed Flownex® SE thermofluid network model.

Imported Libraries

A Flownex.py library is imported, which is provided by Flownex® SE. The default Flownex.py library was adjusted to include a few additional functions. Pandas and NumPy were also imported to handle data-frames from spreadsheets.

```
1. import Flownex
2. import os
3. import re
4. import clr
5. import pandas as pd
6. import numpy as np
7. import time
```

Initiate Flownex Project

Link to the Flownex project and initialise the controller for Flownex® simulation. For the project, the generated inputs were split across 3 PCs, resulting in three separate spreadsheets indexed by the PC number.

```
8. #Start timer to time script run length
9. start = time.time()
10.
11.#Initialise Flownex
12.FlownexSE = Flownex.LaunchApplication()
13.#FSEProject = FlownexSE.Project #(Used for already open Flownex Projects)
14.
15.#Link to the Flownex Project
16.FSEProject = Flownex.OpenProject(FlownexSE,
17.os.path.dirname(os.path.realpath(__file__))+ '\\Your Flownex Project Here.proj')
18.
19.Controller = Flownex.SetupSimulationController(FSEProject)
20.Events = Flownex.SetupSimulationEvents(FSEProject)
21.
22.#Assign Street parameters, number of condenser cells,number of mixed cells, and g
   uess values for steam flow rates for dephlegmators
23.N_Streets = 8
24.N_C = 6
25.N_M = 2
26.m_steam_d_guess = 3
27.
28.#Assign PC number and read in inputs from spreadsheet
29.PC_Number = 1
30.inputs = pd.read_excel('Inputs\\AmbientAir_\\AmbientAirDOE_Split_'+str(PC_Number)
   +'.xlsx')
```

```
31.
32. #Number of input samples
33. n_inputs = inputs.count()[0]
```

Initiate Main Loop for each input sample

The main loop was used to loop through each input from the spreadsheet. The loop contained pre-processing, which assigned the inputs to the Flownex® model, solving, which ran the Flownex® model for the provided inputs, and postprocessing, to store the outputs from the Flownex® model into a spreadsheet.

```
1.  for n in range(0, n_inputs):
2.      print("Processing data set "+str(n))
3.      ##############################Preprocessing###################################
4.      # For selected input sample, get ambient air
5.      # temperature, inlet steam quality, inlet steam
6.      # flow rate, and backpressure guess.
7.      T_air = inputs.iloc[n][0]
8.      steam_qual_in = inputs.iloc[n][1]
9.      bp_guess = inputs.iloc[n][3]
10.     m_steam_d_guess = 3
11.     steam_flow rate = -inputs.iloc[n][2]
12.
13.     # Assign inputs to Flownex model
14.     for i in range(1, N_Streets + 1):
15.         # For each condenser cell, assign ambient air
16.         # temperature input
17.         for k in range(1, N_C + 1):
18.             Flownex.SetInput(FSEProject, "C" + str(i) + str(k) + "-BC-
    Air", "{Boundary Conditions}Temperature",str(T_air) + " °C")
19.
20.         # For each dephlegmator/mixed cell, assign inlet
21.         # steam flow rate guess, and ambient air temperature
22.         for d in range(1, N_M + 1):
23.             Flownex.SetInput(FSEProject, "M" + str(i) + str(d) + "-
    PID", "{Connectable Outputs}Output",str(m_steam_d_guess))
24.             Flownex.SetInput(FSEProject, "M" + str(i) + str(d) + "-
    PID","{Connectable Inputs,Controller Values}Manual Value", str(m_steam_d_guess))
25.             Flownex.SetInput(FSEProject, "M" + str(i) + str(d) + "-BC-
    Air", "{Boundary Conditions}Temperature",str(T_air) + " °C")
26.
27.     # Assign initial backpressure guess
28.     Flownex.SetInput(FSEProject, "BP-
    PID", "{Connectable Outputs}Output", str(bp_guess))
29.     Flownex.SetInput(FSEProject, "BP-
    PID", "{Connectable Inputs,Controller Values}Manual Value", str(bp_guess))
30.
31.     # Assign input steam quality and steam mass flow rate
32.     Flownex.SetInput(FSEProject, "BP-
    BC", "{Boundary Conditions}Quality", str(steam_qual_in))
33.     Flownex.SetInput(FSEProject, "Steam_MF-
    BC", "{Boundary Conditions}Mass source", str(steam_flowrate))
```

```
34.     Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}SP", str(-steam_flowrate))
35.
36.     # Initialise values for Newton-Rhapson solver.
37.     print("Initial inputs have been set")
38.     N_Iter = 0
39.     N_MaxIter = 10
40.     bp_old = 0.0
41.     target_steam = -steam_flowrate
42.     total_cond = 0.0
43.     total_cond_old = 0.0
44.     function_pres = 0.0
45.     function_oldpres = 0.0
46.     grad_out = 0.0
47.     Delta_Back_Pres = 0.0
48.     Relax = 0.95
49.     EB = 0.0
50.
51.     # ##############################Solving#####################################
52.     while N_Iter < N_MaxIter:
53.         print("Iteration " + str(N_Iter))
54.         N_Iter += 1
55.         if N_Iter ==1:
56.             print("Backpressure is " + str(bp_guess))
57.         if (N_Iter == 1):
58.             Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Outputs}Output", str(bp_guess))
59.             Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}Manual Value", str(bp_guess))
60.         elif (N_Iter == 2):
61.             bp_old = bp
62.             total_cond_old = total_cond
63.             function_oldpres = function_pres
64.             # Assign new BP for second iteration based on
65.             # total condensate
66.             if (total_cond > target_steam):
67.                 Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Outputs}Output", str(bp * 0.96))
68.                 Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}Manual Value",str(bp * 0.96))
69.                 print("Backpressure is " + str(bp* 0.96))
70.             else:
71.                 Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Outputs}Output", str(bp * 1.05))
72.                 Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}Manual Value",str(bp * 1.05))
73.                 print("Backpressure is " + str(bp* 1.05))
74.         # If on third iteration or more, calculate new
75.         # backpressure using Newton-Rhapson method.
76.         else:
77.             grad_out = (function_pres - function_oldpres) / (bp - bp_old)
78.             bp_old = bp
79.             total_cond_old = total_cond
80.             function_oldpres = function_pres
81.             Delta_Back_Pres = -function_pres / grad_out
82.             bp = bp + Relax * Delta_Back_Pres
83.             print("Backpressure is " + str(bp))
```

```
84.             Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Outputs}Output", str(bp))
85.             Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}Manual Value", str(bp))
86.
87.         # Initiate steady-state solve in Flownex, wait 5s
88.         # after to ensure fully solved
89.         print("Solving Steady-state")
90.         Flownex.SolveSteadyState(Controller)
91.         time.sleep(5)
92.
93.         # Check if steady-state solve is starting and if it
94.         # is in progress
95.         while (FSEProject.State == 1):
96.             # print("Starting Steady-state...")
97.             if FSEProject.State != 1:
98.                 break
99.         while FSEProject.State == 4:
100.             # print("Solving Steady-state")
101.             if FSEProject.State != 4:
102.                 break
103.           # If an error arises while solving, loop is broken.
104.           if FSEProject.HasErrors == True:
105.               print("Error")
106.               break
107.
108.           # Find total energy imbalance across all
109.           # dephlegmators
110.           for i in range(1, N_Streets + 1):
111.               for d in range(1, N_M + 1):
112.                   EB += np.abs(float(Flownex.GetOutput(FSEProject, "M" + str(i) +
    str(d) + "-PID","{Connectable Inputs,Controller Values}PV")))
113.           print("The total energy imbalance is: " + str(EB))
114.
115.           # Solve transient to minimise energy imbalance
116.           Flownex.StepTransient(Controller)
117.           while FSEProject.State == 0 or FSEProject.State==2:
118.               if FSEProject.State !=0 or FSEProject.State!=2:
119.                   break
120.           iter_transient = 0
121.
122.           # While energy balance is still greater than 0.1 kW,
123.           # keep stepping transiently
124.           while EB > 0.1:
125.               iter_transient += 1
126.               EB = 0
127.               # Calculate energy imbalance
128.               for i in range(1, N_Streets + 1):
129.                   for d in range(1, N_M + 1):
130.                       EB += np.abs(float(Flownex.GetOutput(FSEProject, "M" + str(
   i) + str(d) + "-PID","{Connectable Inputs,Controller Values}PV")))
131.               Flownex.StepTransient(Controller)
132.               while FSEProject.State == 0 or FSEProject.State == 2:
133.                   if FSEProject.State != 0 or FSEProject.State != 2:
134.                       break
135.
136.               # If error arises, break loop
137.               if FSEProject.HasErrors == True:
```

```
                print("Error")
                break
            print("The total energy imbalance is: " + str(EB))

        time.sleep(5)

        # Solve steady-state once transient run is completed
        print("Solving Steady-state")
        Flownex.SolveSteadyState(Controller)
        time.sleep(5)
        while (FSEProject.State == 1):
            # print("Starting Steady-state...")
            if FSEProject.State != 1:
                break
        while FSEProject.State == 4:
            # print("Solving Steady-state")
            if FSEProject.State != 4:
                break
        if FSEProject.HasErrors == True:
            print("Error")
            break

        # Calculate total condensate flowing out of ACC
        # system
        total_cond = float(Flownex.GetOutput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}PV"))
        bp = float(Flownex.GetOutput(FSEProject, "BP-
   PID", "{Connectable Outputs}Output"))
        function_pres = total_cond - target_steam

        # If backpressure is within convergence, stop loop.
        if abs(function_pres) < 0.08:

            break

    # If an error has come up during steady-state or
    # transient, the code attempts to try again to solve, or
    # else stops the solve and loads in the next input
    # sample
    if FSEProject.HasErrors == True:

        # Reverts Flownex model back to a working state with
        # known values
        print("Reverting back to working state")
        Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Outputs}Output", str(14.79874453))
        Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}Manual Value", str(14.79874453))
        Flownex.SetInput(FSEProject, "BP-
   BC", "{Boundary Conditions}Quality", str(0.9245))
        Flownex.SetInput(FSEProject, "Steam_MF-
   BC", "{Boundary Conditions}Mass source", str(-422))
        Flownex.SetInput(FSEProject, "BP-
   PID", "{Connectable Inputs,Controller Values}SP", str(422))
        for i in range(1, N_Streets + 1):
            for k in range(1, N_C + 1):
                Flownex.SetInput(FSEProject, "C" + str(i) + str(k) + "-BC-
   Air", "{Boundary Conditions}Temperature",str(27) + " °C")
```

```
188.             for d in range(1, N_M + 1):
189.                 Flownex.SetInput(FSEProject, "M" + str(i) + str(d) + "-
    PID", "{Connectable Outputs}Output",str(m_steam_d_guess))
190.                 Flownex.SetInput(FSEProject, "M" + str(i) + str(d) + "-
    PID","{Connectable Inputs,Controller Values}Manual Value", str(m_steam_d_guess))
191.                 Flownex.SetInput(FSEProject, "M" + str(i) + str(d) + "-BC-
    Air", "{Boundary Conditions}Temperature",str(27) + " °C")
192.         iter_error = 0
193.
194.         # Loop attempts to solve with known values to get
195.         # the network model back into a working state
196.         while(FSEProject.HasErrors == True):
197.             Flownex.SolveSteadyState(Controller)
198.             time.sleep(5)
199.             while (FSEProject.State == 1):
200.                 # print("Starting Steady-state...")
201.                 if FSEProject.State != 1:
202.                     break
203.             while FSEProject.State == 4:
204.                 # print("Solving Steady-state")
205.                 if FSEProject.State != 4:
206.                     break
207.             if FSEProject.HasErrors == True:
208.                 iter_error += 1
209.                 print("Error, trying Steady-state again")
210.                 # If solving does not work still, reload
211.                 # entire project back to a working state.
212.                 if iter_error >= 4:
213.                     print("Solving still does not work, so reload the entire pr
    oject")
214.                     Flownex.Disconnect(Controller)
215.                     FlownexSE.CloseProject()
216.                     FSEProject = Flownex.OpenProject(FlownexSE, os.path.dirname
    (os.path.realpath(__file__)) + '\\Your Project.proj')
217.                     Controller = Flownex.SetupSimulationController(FSEProject)
218.                     Events = Flownex.SetupSimulationEvents(FSEProject)
219.                     print("Reverting back to working state")
220.                     Flownex.SetInput(FSEProject, "BP-
    PID", "{Connectable Outputs}Output", str(14.79874453))
221.                     Flownex.SetInput(FSEProject, "BP-
    PID", "{Connectable Inputs,Controller Values}Manual Value",str(14.79874453))
222.                     Flownex.SetInput(FSEProject, "BP-
    BC", "{Boundary Conditions}Quality", str(0.9245))
223.                     Flownex.SetInput(FSEProject, "Steam_MF-
    BC", "{Boundary Conditions}Mass source", str(-422))
224.                     Flownex.SetInput(FSEProject, "BP-
    PID", "{Connectable Inputs,Controller Values}SP", str(422))
225.                     for i in range(1, N_Streets + 1):
226.                         for k in range(1, N_C + 1):
227.                             Flownex.SetInput(FSEProject, "C" + str(i) + str(k)
    + "-BC-Air","{Boundary Conditions}Temperature",str(27) + " °C")
228.                         for d in range(1, N_M + 1):
229.                             Flownex.SetInput(FSEProject, "M" + str(i) + str(d)
    + "-PID", "{Connectable Outputs}Output",str(m_steam_d_guess))
230.                             Flownex.SetInput(FSEProject, "M" + str(i) + str(d)
    + "-
    PID","{Connectable Inputs,Controller Values}Manual Value", str(m_steam_d_guess))
```

```
231.                                Flownex.SetInput(FSEProject, "M" + str(i) + str(d)
    + "-BC-Air","{Boundary Conditions}Temperature",str(27) + " °C")
232.
233.                         Flownex.SolveSteadyState(Controller)
234.                         time.sleep(5)
235.                         while (FSEProject.State == 1):
236.                             # print("Starting Steady-state...")
237.                             if FSEProject.State != 1:
238.                                 break
239.                         while FSEProject.State == 4:
240.                             # print("Solving Steady-state")
241.                             if FSEProject.State != 4:
242.                                 break
243.                         iter_error =0
244.                 else:
245.                     break
246.             print("Back to working condition")
247.             print("Skipping to next dataset")
248.             continue
249.
250.         # ##############################Postprocessing##############################
    ####
251.         # Write Results to Spreadsheet Initailise lists for each
252.         # condenser cell and mixed cell parameter
253.         Row1HeatRej_C = []
254.         Row2HeatRej_C = []
255.         Row1InletPresDrop_C = []
256.         Row2InletPresDrop_C = []
257.         Row1HEPresDrop_C = []
258.         Row2HEPresDrop_C = []
259.         Row1MF_C = []
260.         Row2MF_C = []
261.         Row1ExQual_C = []
262.         Row2ExQual_C = []
263.         Row1HTCCond_C = []
264.         Row2HTCCond_C = []
265.         Row1HTCAir_C = []
266.         Row2HTCAir_C = []
267.         FanPower_C = []
268.         FanEff_C = []
269.         AirVolRate_C = []
270.         AirMassFlow_C = []
271.
272.         Row1HeatRej_M_C = []
273.         Row2HeatRej_M_C = []
274.         Row1HeatRej_M_D = []
275.         Row2HeatRej_M_D = []
276.         Row1InletPresDrop_M_C = []
277.         Row2InletPresDrop_M_C = []
278.         Row1HEPresDrop_M_C = []
279.         Row2HEPresDrop_M_C = []
280.         PresDrop_M_D = []
281.         Row1MF_M_C = []
282.         Row2MF_M_C = []
283.         MF_M_D = []
284.         Row1ExQual_M_C = []
285.         Row2ExQual_M_C = []
286.         Row1HTCCond_M_C = []
```

```
287.      Row2HTCCond_M_C = []
288.      Row1HTCAir_M_C = []
289.      Row2HTCAir_M_C = []
290.      Row1HTCCond_M_D = []
291.      Row2HTCCond_M_D = []
292.      Row1HTCAir_M_D = []
293.      Row2HTCAir_M_D = []
294.      FanPower_M = []
295.      FanEff_M = []
296.      AirVolRate_M = []
297.      AirMassFlow_M_C = []
298.      AirMassFlow_M_D = []
299.
300.      CondIndex = []
301.      MixedIndex = []
302.
303.      # Loop through each cell to get the parameter and store
304.      # it in the relevant list above
305.      for i in range(1, N_Streets + 1):
306.          for k in range(1, N_C + 1):
307.              CondIndex.append('C' + str(i) + str(k))
308.              Row1HeatRej_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.G
    etOutput(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}Total heat transfer'))
    )))
309.
310.              Row2HeatRej_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.G
    etOutput(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}Total heat transfer'))
    )))
311.
312.              Row1InletPresDrop_C.append(float(''.join(re.findall(r"\d+\.\d+",Flo
    wnex.GetOutput(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}Inlet Pressure d
    rop'))))))
313.
314.              Row2InletPresDrop_C.append(float(''.join(re.findall(r"\d+\.\d+",Flo
    wnex.GetOutput(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}Inlet Pressure d
    rop'))))))
315.
316.              Row1HEPresDrop_C.append(float(''.join(re.findall(r"\d+\.\d+",Flowne
    x.GetOutput(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}HE Pressure drop'))
    )))
317.
318.              Row2HEPresDrop_C.append(float(''.join(re.findall(r"\d+\.\d+",Flowne
    x.GetOutput(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}HE Pressure drop'))
    )))
319.
320.              Row1MF_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetOut
    put(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}Total mass flow'))))))
321.
322.              Row2MF_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetOut
    put(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}Total mass flow'))))))
323.
324.              Row1ExQual_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
    tOutput(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}Outlet Quality'))))))
325.
326.              Row2ExQual_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
    tOutput(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}Quality'))))))
327.
```

```
328.             Row1HTCCond_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.G
    etOutput(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}HTC Condensate'))))
329.
330.             Row2HTCCond_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.G
    etOutput(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}HTC Condensate'))))
331.
332.             Row1HTCAir_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
    tOutput(FSEProject, 'C' + str(i) + str(k),'{Row 1 Results}HTC Air '))))
333.
334.             Row2HTCAir_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
    tOutput(FSEProject, 'C' + str(i) + str(k),'{Row 2 Results}HTC Air'))))
335.
336.             FanPower_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetO
    utput(FSEProject, 'C' + str(i) + str(k),'{Fan Results}Power (W)'))))
337.
338.             FanEff_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetOut
    put(FSEProject, 'C' + str(i) + str(k),'{Fan Results}Efficiency'))))
339.
340.             AirVolRate_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
    tOutput(FSEProject, 'C' + str(i) + str(k),'{Fan Results}Air Volume Flow Rate'))))
    )
341.
342.             AirMassFlow_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.G
    etOutput(FSEProject, 'C' + str(i) + str(k),'{Air Results}Air Mass Flow'))))
343.
344.         for d in range(1, N_M + 1):
345.             MixedIndex.append('M' + str(i) + str(d))
346.             Row1HeatRej_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
    .GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results}Row 1 Heat Transfe
    r'))))
347.
348.             Row2HeatRej_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
    .GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results}Row 2 Heat Transfe
    r'))))
349.
350.             Row1HeatRej_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
    .GetOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Row 1 Heat Transfe
    r D'))))
351.
352.             Row2HeatRej_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
    .GetOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Row 2 Heat Transfe
    r D'))))
353.
354.             Row1InletPresDrop_M_C.append(float(''.join(re.findall(r"\d+\.\d+",F
    lownex.GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 1}Inlet
    Pressure drop'))))
355.
356.             Row2InletPresDrop_M_C.append(float(''.join(re.findall(r"\d+\.\d+",F
    lownex.GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 2}Inlet
    Pressure drop'))))
357.
358.             Row1HEPresDrop_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flow
    nex.GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 1}Exit Pres
    sure drop'))))
359.
360.             Row2HEPresDrop_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flow
    nex.GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 2}Exit Pres
    sure drop'))))
```

```
361.
362.             PresDrop_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
   tOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Pressure drop'))))
363.
364.             Row1MF_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetO
   utput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 1}Total mass flow')
   ))))
365.
366.             Row2MF_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetO
   utput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 2}Total mass flow')
   ))))
367.
368.             MF_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetOutpu
   t(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Total mass flow'))))
369.
370.             Row1ExQual_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.
   GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 1}Quality'))))
371.
372.             Row2ExQual_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.
   GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 2}Quality'))))
373.
374.             Row1HTCCond_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
   .GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 1}HTC Condensa
   te'))))
375.
376.             Row2HTCCond_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
   .GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 2}HTC Condensa
   te'))))
377.
378.             Row1HTCAir_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.
   GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 1}HTC Air'))))
379.
380.             Row2HTCAir_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.
   GetOutput(FSEProject, 'M' + str(i) + str(d),'{K Steam Results,Row 2}HTC Air'))))
381.
382.             Row1HTCCond_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
   .GetOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Row 1 HTC Condensa
   te'))))
383.
384.             Row2HTCCond_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
   .GetOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Row 2 HTC Condensa
   te'))))
385.
386.             Row1HTCAir_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.
   GetOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Row 1 HTC Air'))))
387.
388.             Row2HTCAir_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.
   GetOutput(FSEProject, 'M' + str(i) + str(d),'{D Steam Results}Row 2 HTC Air'))))
389.
390.             FanPower_M.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetO
   utput(FSEProject, 'M' + str(i) + str(d),'{Fan Results}Power'))))
391.
392.             FanEff_M.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.GetOut
   put(FSEProject, 'M' + str(i) + str(d), '{Fan Results}Efficiency'))))
393.
394.             AirVolRate_M.append(float(''.join(re.findall(r"\d+\.\d+",Flownex.Ge
   tOutput(FSEProject, 'M' + str(i) + str(d),'{Air Results}Total Air Volume Flow Rat
   e'))))
```

```
395.
396.             AirMassFlow_M_C.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
   .GetOutput(FSEProject, 'M' + str(i) + str(d),'{Air Results}K Air Mass Flow')))))
397.
398.             AirMassFlow_M_D.append(float(''.join(re.findall(r"\d+\.\d+",Flownex
   .GetOutput(FSEProject, 'M' + str(i) + str(d),'{Air Results}D Air Mass Flow')))))
399.
400.     misc_res = []
401.     BP_fin = float(''.join(re.findall(r"\d+\.\d+", Flownex.GetOutput(FSEProject
   , 'BP-BC', '{Boundary Conditions}Pressure'))))
402.
403.     SDD_drop = float(''.join(re.findall(r"\d+\.\d+", Flownex.GetOutput(FSEProje
   ct, 'Steam Duct Loss - 3008', '{Duct Loss}Pressure drop'))))
404.
405.     InQual = float(''.join(re.findall(r"\d+\.\d+", Flownex.GetOutput(FSEProject
   , 'BP-BC', '{Boundary Conditions}Quality'))))
406.
407.     InMF = float(''.join(re.findall(r"\d+\d+", Flownex.GetOutput(FSEProject, 'S
   team_MF-BC', '{Boundary Conditions}Mass source'))))
408.
409.     AirTemp =float(''.join(re.findall(r"\d+\.\d+", Flownex.GetOutput(FSEProject
   , 'C11-BC-Air', '{Boundary Conditions}Temperature'))))
410.
411.     misc_res.append(BP_fin)
412.     misc_res.append(SDD_drop)
413.     misc_res.append(InQual)
414.     misc_res.append(InMF)
415.     misc_res.append(AirTemp)
416.
417.     # Store parameters in a pandas dataframe
418.     cond_df = pd.DataFrame(list(zip(Row1HeatRej_C, Row2HeatRej_C, Row1InletPres
   Drop_C, Row2InletPresDrop_C,Row1HEPresDrop_C, Row2HEPresDrop_C, Row1MF_C, Row2MF_
   C, Row1ExQual_C, Row2ExQual_C,Row1HTCCond_C, Row2HTCCond_C, Row1HTCAir_C, Row2HTC
   Air_C, FanPower_C, FanEff_C, AirVolRate_C, AirMassFlow_C)),columns=['Row 1 Heat R
   ej', 'Row 2 Heat Rej', 'Row 1 Inlet Pres Drop', 'Row 2 Inlet Pres Drop','Row 1 HE
    Pres Drop', 'Row 2 HE Pres Drop', 'Row 1 MassFlowRate', 'Row 2 MassFlowRate','Ro
   w 1 Exit Qual', 'Row 2 Exit Qual', 'Row 1 HTC Cond', 'Row 2 HTC Cond', 'Row 1 HTC
    Air', 'Row 2 HTC Air','Fan Power', 'Fan Efficiency', 'Air Vol Flow Rate', 'Air M
   ass Flow Rate'],index=CondIndex)
419.     mixed_df = pd.DataFrame(list(zip(Row1HeatRej_M_C, Row2HeatRej_M_C, Row1Heat
   Rej_M_D, Row2HeatRej_M_D, Row1InletPresDrop_M_C, Row2InletPresDrop_M_C,Row1HEPres
   Drop_M_C, Row2HEPresDrop_M_C, PresDrop_M_D, Row1MF_M_C, Row2MF_M_C, MF_M_D, Row1E
   xQual_M_C, Row2ExQual_M_C,Row1HTCCond_M_C, Row2HTCCond_M_C, Row1HTCAir_M_C, Row2H
   TCAir_M_C, Row1HTCCond_M_D, Row2HTCCond_M_D, Row1HTCAir_M_D, Row2HTCAir_M_D,FanPo
   wer_M, FanEff_M, AirVolRate_M, AirMassFlow_M_C, AirMassFlow_M_D)),
420.     columns=['C Row 1 Heat Rej', 'C Row 2 Heat Rej', 'D Row 1 Heat Rej', 'D Row
    2 Heat Rej', 'C Row 1 Inlet Pres Drop', 'C Row 2 Inlet Pres Drop','C Row 1 HE Pr
   es Drop', 'C Row 2 HE Pres Drop', 'D Pres Drop', 'C Row 1 MF', 'C Row 2 MF', 'D M
   F', 'C Row 1 Exit Qual', 'C Row 2 Exit Qual','C Row 1 HTC Cond', 'C Row 2 HTC Con
   d', 'C Row 1 HTC Air', 'C Row 2 HTC Air','D Row 1 HTC Cond', 'D Row 2 HTC Cond',
   'D Row 1 HTC Air', 'D Row 2 HTC Air', 'Fan Power', 'Fan Efficiency', 'Air Vol Flo
   w Rate','C Air Mass Flow', 'D Air Mass Flow'],index=MixedIndex)
421.
422.     misc_df = pd.DataFrame(misc_res, index=['Final Backpressure', 'SDD Pressure
    Drop (Pa)', 'Inlet Quality', ' Total Steam Mass Flow In', 'Inlet Air Temps'])
423.
424.     # Write dataframe with outputs into a spreadsheet
```

```
425.     with pd.ExcelWriter('Results\\AmbientAirRes_4\\PC '+str(PC_Number)+'\\Ambie
   ntAirRes_'+str(PC_Number)+'_' +str(n)+'.xlsx') as writer:
426.         cond_df.to_excel(writer, sheet_name='Condenser Cells')
427.         mixed_df.to_excel(writer, sheet_name='Mixed Cells')
428.         misc_df.to_excel(writer, sheet_name='Misc Results')
429.
430. #Once the main loop has completed, disconnect the controller and save the proje
   ct.
431. Flownex.Disconnect(Controller)
432. Flownex.SaveProject(FlownexSE)
433.
434. #Script wall time
435. print("It took", time.time()-start, " seconds to run these simulations.")
```

Appendix C. Machine Learning Code Listing

C.1 Regression MLP Network

This notebook includes the relevant code used to develop the regression MLP network.

Imported Libraries

PyTorch was used as the fundamental framework for ML development. Additionally, Skorch was used to integrate Scikit-Learn with the developed PyTorch model. Scikit-learn included all the relevant metrics and pre- and post-processing functions.

```
1. import torch
2. import torch.nn as nn
3. import torch.nn.functional as F
4. from torch.utils.data import DataLoader
5. from torch.utils.data import Dataset, TensorDataset
6. import numpy as np
7. import pandas as pd
8. import skorch
9. from skorch import NeuralNetRegressor
10. from sklearn.model_selection import train_test_split
11. from sklearn.preprocessing import MinMaxScaler
12. from sklearn.metrics import mean_squared_error as MSE
13. from sklearn.externals import joblib
```

Pre-processing

Pre-processing involved reading in the combined dataset from a spreadsheet, separating the inputs and outputs, and scaling the data using a min-max scaler. A separate min-max scaler was used to scale the inputs, and another for the outputs. Scikit-learn's "train_test_split" function was used to split the dataset into an 80/20 split for training and testing respectively.

```
14. #Set device to Cuda to use GPU
15. device = torch.device("cuda")
16.
17. #Set seeds to ensure reproducibility
18. torch.manual_seed(2020)
19. np.random.seed(2020)
20.
21. #Read in data from spreadsheet
22. data = pd.read_excel('Combined Results.xlsx', index = False)
23.
24. #Get inputs from data and scale
25. inputs_proc = data.iloc[:,0:6]
26. scaler_in_minmax = MinMaxScaler()
27. scaler_out_minmax = MinMaxScaler()
28. inputs_proc = scaler_in_minmax.fit_transform(inputs_proc)
29.
```

```
30. #Get outputs from data and scale
31. outputs_proc_no_in = data.drop(data.iloc[:, 0:6], axis = 1)
32.
33. #Drop total heat rejection rates as these columns are duplicates
34. outputs_proc_no_tot =  outputs_proc_no_in.drop(['Total Heat Rej (MW)', 'Total Air
    Vol Flow Rate', 'Total Fan Power (kW)'], axis = 1)
35. outputs_proc = outputs_proc_no_tot
36. outputs_proc = scaler_out_minmax.fit_transform(outputs_proc)
37.
38. #Dump scalers to save and use in deployment
39. joblib.dump(scaler_in_minmax, 'Reg_scaler_in.save')
40. joblib.dump(scaler_out_minmax, 'Reg_scaler_out.save')
41.
42. #Use Scikit-learn to split for train/test 80/20
43. X_train, X_test, y_train, y_test = train_test_split(inputs_proc, outputs_proc, te
   st_size = 0.2)
44. X_train = X_train.astype(np.float32)
45. X_test = X_test.astype(np.float32)
46. y_train = y_train.astype(np.float32)
47. y_test = y_test.astype(np.float32)
48.
49. #Define input and output vector dimensions
50. input_size = 6
51. output_size = 7
```

Regression MLP Network Setup

The regression MLP network was then developed using the architecture selected for the final regression network.

```
52. class RegNet(nn.Module):
53.     def __init__(self, input_size,
54.     hidden_size,
55.     output_size):
56.         super(RegNet, self).__init__()
57.
58.         #Input layer
59.         self.fc1 = nn.Linear(input_size, hidden_size)
60.
61.         #Hidden Layers (all hidden layers had the same number of neurons,
62.         #therefore, a single fully connected layer function was used)
63.         self.fcx = nn.Linear(hidden_size, hidden_size)
64.
65.         #Output Layer
66.         self.out = nn.Linear(hidden_size, output_size)
67.
68.     #Def forward propagation
69.     def forward(self,x):
70.         #ReLU on input layer
71.         x = F.relu(self.fc1(x))
72.
73.         #ReLU on hidden four hidden layers
74.         x = F.relu(self.fcx(x))
75.         x = F.relu(self.fcx(x))
76.         x = F.relu(self.fcx(x))
77.         x = F.relu(self.fcx(x))
```

```
78.
79.         #Linear output layer
80.         x = self.out(x)
81.         return x
```

Training

Skorch integration and metrics:

```
82. #Define MSE loss criterion
83. criterion = nn.MSELoss(reduction = 'mean')
84.
85. #Skorch NeuralNetRegressor for regression neural net
86. net_prac = NeuralNetRegressor(
87.     #Uses previously defined PyTorch RegNet, specify network hyperparameters
88.     module = RegNet,
89.     module__input_size = input_size,
90.     module__hidden_size =4096,
91.     module__output_size = output_size,
92.     criterion = nn.MSELoss,
93.     criterion__reduction='mean',
94.     optimizer = torch.optim.Adam,
95.     max_epochs = 500,
96.     lr = 0.0001,
97.     batch_size = 64,
98.     device = 'cuda',
99.     verbose = 1
100. )
101.
102. #Train neural net using training dataset
103. net_prac.fit(X_train, y_train)
```

Test Set Evaluation

Final model is used to provide predictions using test set:

```
104. print(net_prac)
105.
106. #Evalute outputs using regression network on test set
107. y_pred = net_prac.validation_step(X_test, y_test)
108. print(y_pred['loss'])
109.
110. #Find actual outputs and predicted outputs inverse-passed through min-
     max scaler
111. y_test_actual = scaler_out_minmax.inverse_transform(y_test)
112. y_pred_actual = scaler_out_minmax.inverse_transform(y_pred['y_pred'].cpu())
113.
114. #Find average MSE on test set
115. print(MSE(y_test_actual, y_pred_actual)**(1/2))
116.
117. #Dump final model to use
118. joblib.dump(net_prac, 'RegFinal.pkl')
```

Gridsearch/Cross-Validation

If using a grid-search or cross-validation, the code snippet below can be used. The grid-search can be used to search a coarse grid of hyperparameters through adding values in the lists below for the defined hyperparameters. If a single network needs to be tested using cross-validation, the single parameters should be defined in the 'params' section below.

```
#If using a gridsearch with 6-fold cross validation:
#Define parameters used in gridsearch in lists
params = {
    'lr' : [1E-4],
    'module__n_hidlayers' : [5],
    'module__hidden_size' : [4096],
    'max_epochs' : [500],
    'batch_size' :[64]
}

#Gridsearch with cross-validation using Skorch and Scikit-Learn
gs = GridSearchCV(net_GS, params, refit = True, cv = 6, scoring = 'neg_mean_squared_error', n_jobs= 1, verbose= 2)

#Fit using training set for all developed networks
gs.fit(X_train, y_train)

#Define code report which reports best performing networks using validation score (top 3)
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.8f} (std: {1:.8f})".format(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}".format(results['params'][candidate]))
            print("")
report(gs.cv_results_, 10)

#Use best network on test set
print(gs.best_estimator_)
y_pred = gs.best_estimator_.validation_step(X_test, y_test)
print(y_pred['loss'])
y_test_actual = scaler_out_minmax.inverse_transform(y_test)
y_pred_actual = scaler_out_minmax.inverse_transform(y_pred['y_pred'].cpu())
print(MSE(y_test_actual, y_pred_actual)**(1/2))
```

C.2 Binary Classifier MLP Network

This notebook includes the relevant code used to develop the binary classifier MLP network.

Imported Libraries

PyTorch was used as the fundamental framework for ML development. Additionally, Skorch was used to integrate Scikit-Learn with the developed PyTorch model. Scikit-learn included all the relevant metrics and pre- and post-processing functions.

```
1. import torch
2. import torch.nn as nn
3. import torch.nn.functional as F
4. import numpy as np
5. import pandas as pd
6. import time
7. import skorch
8. from skorch import NeuralNetRegressor
9. from sklearn.model_selection import GridSearchCV
10. from sklearn.model_selection import train_test_split
11. from sklearn.preprocessing import MinMaxScaler
12. from sklearn.metrics import mean_squared_error as MSE
13. from sklearn.metrics import log_loss as ll
14. from sklearn.externals import joblib
15. from sklearn.metrics import accuracy_score
```

Pre-processing

Pre-processing involved reading in the combined dataset from a spreadsheet, separating the inputs and outputs, and scaling the data using a min-max scaler. A separate min-max scaler was used to scale the inputs, and another for the outputs. Scikit-learn's train_test_split function was used to split the dataset into a 80/20 split for training and testing respectively.

```
16. #Set device to Cuda to use GPU
17. device = torch.device("cuda")
18.
19. #Set seeds to ensure reproducibility
20. torch.manual_seed(2020)
21. np.random.seed(2020)
22.
23. #Read in data from spreadsheet
24. data = pd.read_excel('Combined Results.xlsx', index = False)
25.
26. #Get inputs from data and scale
27. inputs_proc = data.iloc[:,0:6]
28. scaler_in_minmax = MinMaxScaler()
29. scaler_out_minmax = MinMaxScaler()
30. inputs_proc = scaler_in_minmax.fit_transform(inputs_proc)
31.
32. #Get outputs from data and scale
33. outputs_proc_no_in = data.drop(data.iloc[:, 0:6], axis = 1)
```

```
34.
35. #Drop total heat rejection rates as these columns are duplicates
36. outputs_proc_no_tot =  outputs_proc_no_in.drop(['Total Heat Rej (MW)', 'Total Air
    Vol Flow Rate', 'Total Fan Power (kW)'], axis = 1)
37. outputs_proc = outputs_proc_no_tot
38. outputs_proc = scaler_out_minmax.fit_transform(outputs_proc)
39.
40. #Dump scalers to save and use in deployment
41. joblib.dump(scaler_in_minmax, 'Reg_scaler_in.save')
42. joblib.dump(scaler_out_minmax, 'Reg_scaler_out.save')
43.
44. #Use Scikit-learn to split for train/test 80/20
45. X_train, X_test, y_train, y_test = train_test_split(inputs_proc, outputs_proc, te
    st_size = 0.2)
46. X_train = X_train.astype(np.float32)
47. X_test = X_test.astype(np.float32)
48. y_train = y_train.astype(np.float32)
49. y_test = y_test.astype(np.float32)
50.
51. #Define input and output vector dimensions, 1 output for Binary Classifier
52. input_size = 6
53. output_size = 1
```

Binary Classifier MLP Network Setup

The regression MLP network was then developed using the architecture selected for the final binary classifier network.

```
54. class Net(nn.Module):
55.
56.     def __init__(self, input_size,
57.     hidden_size,
58.      output_size):
59.
60.         super(Net, self).__init__()
61.         #Input layer
62.         self.fc1 = nn.Linear(input_size, hidden_size)
63.
64.         #Hidden Layers (all hidden layers had the same number of neurons,
65.         #therefore a single fully connected layer function was used)
66.         self.fcx = nn.Linear(hidden_size, hidden_size)
67.
68.         #Output Layer
69.         self.output = nn.Linear(hidden_size, output_size)
70.
71.     def forward(self,x):
72.         #ReLU on input layer
73.         x = F.relu(self.fc1(x))
74.
75.         #ReLU on hidden layers
76.         x = F.relu(self.fcx(x))
77.         x = F.relu(self.fcx(x))
78.         x = F.relu(self.fcx(x))
79.
80.         #Sigmoid on output layer
81.         x = torch.sigmoid(self.output(x))
```

```
82.
83.         return x
```

Training

Skorch integration and metrics:

```
84. #Binary Cross-entropy for loss criterion
85. criterion = nn.BCELoss()
86. #Callbacks to show accuracy metric
87. train_acc =skorch.callbacks.EpochScoring(scoring='accuracy', on_train=True,
88.                          name='train_acc', lower_is_better=False)
89.
90. callbacks = [train_acc]
91.
92. #Skorch neural net binary classifier integration
93. net_Pred = skorch.classifier.NeuralNetBinaryClassifier(
94.     module = Net,
95.     module__input_size = input_size,
96.     module__hidden_size = 512,
97.     module__output_size = output_size,
98.     criterion= nn.BCELoss,
99.     optimizer = torch.optim.Adam,
100.      max_epochs = 81,
101.      lr = 0.001,
102.      batch_size = 64,
103.      device = 'cuda',
104.      verbose = 1,
105.      callbacks = [train_acc]
106.
107. )
108.
109. #Fit on training dataset
110. net_Pred.fit(X_train, y_train)
```

Test Set Evaluation

Final model is used to provide predictions using test set:

```
111. # Try with eval step first
112. print(net_Pred)
113. y_pred = net_Pred.validation_step(X_test, y_test)
114. print(y_pred['loss'])
115.
116. y_test_actual = scaler_out_minmax.inverse_transform(y_test)
117. y_pred_actual = scaler_out_minmax.inverse_transform(y_pred['y_pred'].cpu().resh
     ape(-1, 1)).round()
118.
119. #Find logloss on test set
120. print(ll(y_test_actual, y_pred_actual))
121. #Find classification accuracy
122. print(accuracy_score(y_test_actual, y_pred_actual))
```

Grid-search/Cross-Validation

If using a grid-search or cross-validation, the code snippet below can be used. The grid-search can be used to search a coarse grid of hyperparameters through adding values in the lists below for the defined hyperparameters. If a single network needs to be tested using cross-validation, the single parameters should be defined in the 'params' section below.

```
#If using a gridsearch with 6-fold cross validation:
#Define parameters used in gridsearch in lists
params = {
    'lr' : [1E-4],
    'module__n_hidlayers' : [5],
    'module__hidden_size' : [4096],
    'max_epochs' : [500],
    'batch_size' :[64]
}
#Gridsearch with cross-validation using Skorch and Scikit-Learn
gs = GridSearchCV(net_Pred, params, refit = True, cv = 6, scoring = 'neg_mean_squared_error', n_jobs= 1, verbose= 2)

#Fit using training set for all developed networks
gs.fit(X_train, y_train)

#Define code report which reports best performing networks using validation score (top 3)
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.8f} (std: {1:.8f})".format(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}".format(results['params'][candidate]))
            print("")
report(gs.cv_results_, 10)

#Use best network on test set
print(gs.best_estimator_)
y_pred = gs.best_estimator_.validation_step(X_test, y_test)
print(y_pred['loss'])
y_test_actual = scaler_out_minmax.inverse_transform(y_test)
y_pred_actual = scaler_out_minmax.inverse_transform(y_pred['y_pred'].cpu())
#Find logloss on test set
print(ll(y_test_actual, y_pred_actual))
#Find classification accuracy
print(accuracy_score(y_test_actual, y_pred_actual))
```

Appendix D. Web-App Prototype Development

This manual serves as a guide to using the air-cooled condenser prediction tool developed for a utility-scale ACC system in South Africa. The prediction tool was developed using a web app through Flask and was based on a surrogate data-driven neural network model. The neural network model was developed based on data generated by the 1-D thermofluid network model of the ACC system under various conditions, as discussed in Chapter 3 and Chapter 5, respectively.

Ideally, the web-app will be run on a server, and a user will be able to connect through a specified local domain (specified when running the main file, app.py). This allows for local access over a network, or a single computer if preferred. This user guide details how to use the already accessible prediction tool, rather than setting it up to run (although that is not difficult with the provided files and installed dependencies). The tool can be accessed via http://accwebtool.af-south-1.elasticbeanstalk.com/ using the username User and the password RAH_Masters_2020.

The data-driven model has limitations because of the range of operating conditions the model has been trained on. Neural networks do not extrapolate very well, and therefore it was essential to define a clear scope for which this prediction tool could be used. The following sections will expand on the scope and use cases for this prediction tool to ensure a reliable prediction. However, it must be noted that this model alone will not be accurate enough to capture performance, especially under windy conditions. An additional study on the effect of crosswind speed on air mass flow rate reduction for the ACC system must be conducted to ensure validation on the air mass flow rate predicted by this model. Consequently, this tool acts as a proof of concept of how a prediction tool could work for an ACC system from the surrogate data-driven model.

The main aim of the web-app prototype development was to explore deployment of machine learning models into a practical environment. The web-app is not meant to be a final fully developed solution, but rather an example of implementation.

D.1 Prediction Tool Scope:

The generated data from the 1-D network model was limited in the range of ambient air temperatures, inlet steam qualities, inlet steam mass flow rates, wind angles, wind speeds, and lastly the number of ACC streets operating. This was explained in detail in Section 5.1.1 and 5.2.1 from the three generated datasets. For inputs, the model takes in inputs and provides outputs, as shown in Table 22.

Firstly, a general scope of inputs was defined in Table 30. The model cannot solve any inputs outside the specified ranges.

Table 30: General input scope (x is input variable)

Ambient Air Temperature $(°C)$	$4 < x < 42$
ACC Inlet Steam Quality	$0.86 < x < 1.0$
ACC Inlet Steam Flow rate (kg / s)	$350 < x < 450$
Wind Angle	Cardinal and Ordinal directions (N, E, S, W) (NE, SE, SW, NW)
Wind Speed (m / s)	$2.5 < x < 7.0$
Number of Streets switched off	0, 1, 2

The tables below then break down these ranges further and go into how the different input variables affect each other. Specific inputs constrained the ranges of others. Table 31 shows the seasonal ambient air temperature operating breakdown (dataset 1). This input set considers no wind effects and works with all ACC streets operating. If using inputs which lie in these conditions, the wind angle needs to be set as "N" and wind speed to be 2.5 m / s as a default setting.

Table 31: Seasonal ambient air temperatures (dataset 1)

Ambient Air Temperature $(°C)$	$12 < x < 42$
ACC Inlet Steam Quality	$0.86 < x < 1.0$
ACC Inlet Steam Flow rate (kg / s)	$350 < x < 450$
Wind Angle	N/A ('N' as default)
Wind Speed (m / s)	N/A (2.5 as default)
Number of Streets switched off	0

Similarly, for dataset 2, which considered one street and two streets switched, Table 32 showed the acceptable ranges used for the model.

Table 32: Winter operating conditions with one street off and two streets off (dataset 2)

	1 Street Off	2 Streets Off
Ambient Air Temperature $(°C)$	$7.5 < x < 24$	$4.5 < x < 24$
ACC Inlet Steam Quality	$0.86 < x < 1.0$	$0.86 < x < 1.0$
ACC Inlet Steam Flow rate (kg / s)	$350 < x < 450$	$350 < x < 450$
Wind Angle	N/A ('N' as default)	N/A ('N' as default)
Wind Speed (m / s)	N/A (2.5 as default)	N/A (2.5 as default)
Number of Streets switched off	1	2

When considering wind effects, Table 33 shows the input scope. For wind speed, a limitation on the maximum wind speed was specified, limited by the maximum fan speed reduction in the generated dataset. A limitation on the lower end at $2.5\ m/s$ was also specified due to the wind speed at design conditions of the ACC system modelled.

Table 33: Wind effects operating range

Ambient Air Temperature $(°C)$	$9 < x < 30$
ACC Inlet Steam Quality	$0.86 < x < 1.0$
ACC Inlet Steam Flow rate (kg / s)	$350 < x < 450$
Wind Angle	N, NE, E, SE, S, SW, W, NW
Wind Speed (m / s)	$2.5 < x < 7.0$
Number of Streets switched off	0

To summarise:

- When not considering wind effects:
    - See Table 31
        - If the specified temperature is lower than the specified range in Table 31, see the first column of Table 32 for one street switched off, and if still does not solve then try switching off another street (maximum of two streets).
- When considering wind effects:
    - See Table 33
- If any of the provided input ranges lie outside the provided scope, the prediction tool will not give reliable results. The binary classifier should flag these input samples and indicate which input samples were non-solvable.

D.2 Using the Prediction Tool

The prediction tool consists of three main sections, seen in Figure 69, which are for “Single Inputs”, “Multiple Inputs”, and “Forecasting”.

Figure 69: Landing page

If predictions for a single set of inputs are required, please click the "Single Input" button. If there are multiple inputs (in an .xlsx or .csv file), use the "Upload Multiple Files" button. Lastly, a prediction based on forecasted weather data at ACC system locality could be requested using the "Forecast" button. The forecasting section was the main focus of the web-app.

Single Inputs

Figure 70 shows the landing page for “Single Inputs”. There are six input boxes for specifying the required inputs.

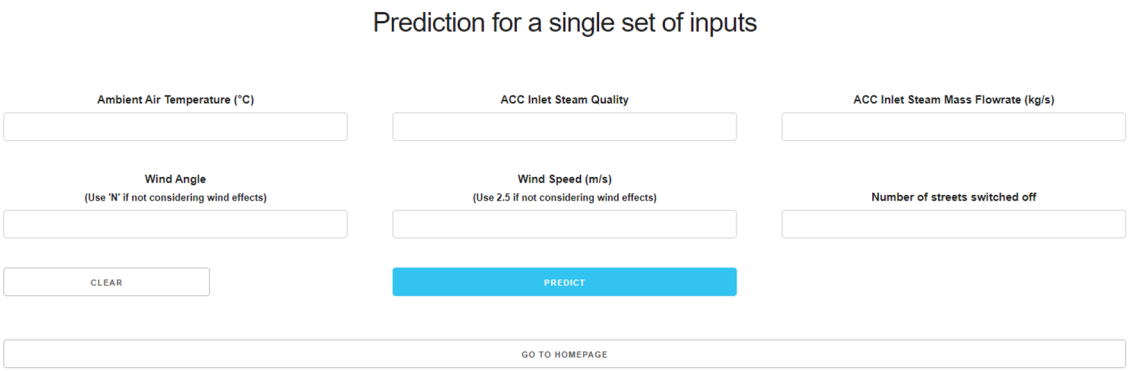


Figure 70: Single inputs landing page

All inputs must be filled out according to Section D.1, or else the user will receive an error: "The provided inputs were out of range of the program and are unsolvable by this model. Please consult the manual for more information". If the provided inputs do not lie in the defined scope, predictions cannot be produced for them by this prediction tool.

Wind angles must be specified as outlined in Table 33; other wind angles will result in "Wind angle(s) has been entered incorrectly. Please revise".

If inputs without considering wind effects were needed, in the Wind Angle box type "N", and for Wind Speed type "2.5".

Once the inputs have been provided in their respective boxes, the 'Predict' button will proceed to run the data-driven surrogate model, producing ACC performance predictions. The user can then copy the results table.

Multiple Inputs

Multiple inputs work the same as the single input section, however, just applies to more than one input, in the form of a spreadsheet or a CSV file.

Upload:

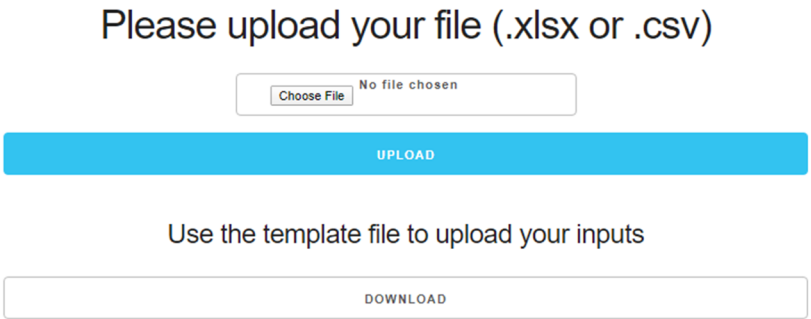


Figure 71: Multiple inputs landing page

Figure 71 shows the landing page for “Multiple Inputs”. The template file must be downloaded and used when specifying multiple inputs. Column titles should be left as provided. User inputs should be specified through multiple rows. The user should save their inputs and then upload them through the dialogue box. Once the file has been selected using the dialogue box, they can then be uploaded.

Processing:

Once uploaded, the next page will display the provided inputs. Any inputs that were unsolvable by the prediction tool will be displayed. If all inputs were specified incorrectly, the user would have to go back and revise their inputs to ensure they are in accordance with Section D.1.

If only a few of the provided inputs are incompatible (either incorrectly specified or out of range/incompatible), the prediction tool will be able to carry on without the incompatible inputs. The user can either fix the incompatible inputs before proceeding or proceed without them by clicking the “Predict” button.

Prediction:

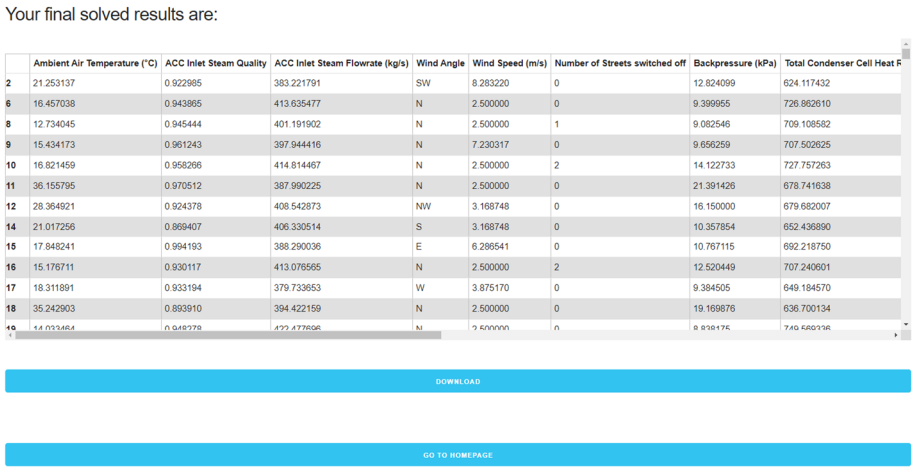


Figure 72: Multiple inputs Prediction page

Figure 72 shows the proceeding prediction page. The user can view the inputs for which predictions were produced. An excel file containing predictions with working inputs, as well as a sheet with incompatible inputs, can be downloaded.

Forecasting

Lastly, the “Forecast” section shows ACC prediction results based on a forecast of ambient air temperatures, wind speeds, and wind angles for the ACC system locality. The weather forecasts are

provided by the Open Weather API [62]. The forecasts were provided at 3-hour intervals for the next five days from the user request. Please be aware that provided forecasts may be incompatible with the prediction tool and some inputs may thus be omitted.

The generated plots were interactive. The user can hover over individual data points in each plot for more insight into specific values. Additionally, legends are also interactive, and clicking on an item in a legend will disable/enable the item in the respective graph. Panning is uniform across all the plots.

Predictions can be downloaded in an excel (.xlsx) file, containing the dates and times, and all results for those intervals. Once again, please note that when considering wind effects, model results require additional validation.

If wind speeds are relatively high and the user would still like predictions based on design conditions (i.e. no wind effects), a “forecast without wind effects” button is located at the bottom of the page. This will take the forecasted weather data (without the wind effects) and predict ACC performance at design conditions. Forecasting without wind effects was implemented to observe what effects wind has on different performance parameters.

This concludes the manual on the web-app prototype developed for the project. The web-app prototype was developed to deploy the data-driven surrogate model, improving the practicality and demonstrating how data-driven surrogate models could potentially be integrated on-site. It is important to note that the web-app prototype developed in this research requires additional validation and enhancement to be a useable condition monitoring tool.